University of Minho

School of Engeneering

Department of Information Systems

Gisela Maria Nogueira Fernandes

Pervasive Data Science applied to the Society of Services

Master's Dissertation

Information Systems Engineering and Management Integrated Master

Work developed with guidance from:

Professor Manuel Filipe Vieira Torres dos Santos

Professor Carlos Filipe da Silva Portela

October 2019

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

LICENÇA CONCEDIDA AOS UTILIZADORES DESTE TRABALHO

ACKNOWLEDGMENTS

Now that it is ending one of the most enthusiastic, overwhelming and simply incredible phases of my life, I want to take this opportunity to thank every single person that shared this path with me and somehow helped me to turn this journey unforgettable.

To my parents, who have hardly worked so I could pursue my dreams and graduate myself with the best conditions possible. For every caring word, for all the support you gave me, never letting me give up. The results I get from this course, and particularly this dissertation are my way of honouring you, and showing you that it was worth it.

To my advisor, Professor Manuel Filipe Santos, and all the knowledge and experience shared across this course.

To my coordinator, Professor Filipe Portela, one of the most admirable persons that I had the pleasure to meet during these years, that became an example of hardworking, commitment, and professionalism. For all the knowledge shared, dedication, and values, that for sure I'll carry to my life.

To all my colleagues and friends that shared the classes but also all the moments out of it, to all our talks, jokes, and argues. I want to particularly thank all the ones with who I had the pleasure to directly work with across the different disciplines, for all the hours we spent cracking our heads together (sometimes just pretending to) that helped me grow as a person and gain a group of friends that I intend to keep close.

To all my friends spread worldwide with who I had the opportunity to share the biggest adventure of my life so far, to my Erasmus crew.

To all my family and friends from Braga that walked by side all the time, earing my complaints even when they did not understand a thing of what I was talking about. Especially to my best friend Sara, to all our great moments, our "study" coffees, and joined panic attacks.

To all my colleagues and friends that I had the pleasure to work with these past few months at IOTech. For all the knowledge shared, all the funny moments, and for welcoming me into this amazing ioFamily. I want to especially thank José, my advisor inside the company, Adriano and Pedro, for all the help they gave me towards this dissertation success.

Last but not least, to this marvellous house that took me in, to the University of Minho and Information Systems Department, and all its members that somehow contributed to my formation.

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

RESUMO

Com o avanço tecnológico que se tem vindo a notar nos últimos anos e, atualmente, com a implementação do conceito *Internet of Things*, é possível observar o enorme crescimento dos volumes de dados recolhidos a cada minuto. Esta realidade levanta uma problemática: "Como podemos processar grandes volumes dados e extrair conhecimento a partir deles em tempo útil?". Este não é um problema fácil de resolver pois muitas vezes não estamos a lidar apenas com grandes volumes de dados, mas também com diferentes tipos dos mesmos, o que torna a problemática ainda mais complexa.

Atualmente, grandes quantidades dos mais variados tipos de dados são geradas. Estes dados por si só não acrescentam qualquer valor às organizações que os recolhem. Porém, quando submetidos a processos de análise, podem ser convertidos em fontes de informação cruciais no centro do negócio. Assim sendo, o foco deste projeto é explorar esta problemática e tentar atribuir-lhe uma solução modular e adaptável a diferentes realidades, com base em tecnologias atuais que permitam ao utilizador aceder à informação onde e quando quiser.

Na primeira fase desta dissertação, foi executada uma pesquisa bibliográfica, assim como, uma revisão da literatura recolhida nessas mesmas fontes, a fim de compreender que soluções já foram propostas e quais são as questões que requerem uma resposta.

Numa segunda fase, foi desenvolvida uma solução, composta por quatro modulos, que passa por submeter os dados a um processo de tratamento (onde estão incluídas onze funções de tratamento, com o objetivo de preencher o modelo multidimensional previamente desenhado) e, posteriormente, desenvolver uma camada OLAP que seja capaz de lidar não só com dados estruturados, mas também dados não estruturados. No final, é possível consultar um conjunto de quatro *dashboards* disponibilizados numa plataforma web que tem como base mais de vinte queries iniciais, e filtros com base numa query dinamica.

Para este caso de estudo e como prova de conceito foi utilizada a empresa IOTech, empresa que disponibilizará os dados necessários para suportar esta dissertação, e com base nos quais foram definidos cinco *Key Performance Indicator*s.

Durante este projeto foram aplicadas diferentes metodologias: *Design Science Research*, no que diz respeito à pesquisa, e SCRUM, no que diz respeito à componente prática.

Palavras-chave: dados não estruturados, bases de dados NoSQL, extração de conhecimento, análises em tempo real, OLAP.

ABSTRACT

With the technological progress that has been happening in the last few years, and now with the actual implementation of the Internet of Things concept, it is possible to observe an enormous amount of data being collected each minute. Well, this brings along a problem: "How can we process such amount of data in order to extract relevant knowledge in useful time?". That's not an easy issue to solve, because most of the time one needs to deal not just with tons but also with different kinds of data, which makes the problem even more complex.

Today, and in an increasing way, huge quantities of the most varied types of data are produced. These data alone do not add value to the organizations that collect them, but when subjected to data analytics processes, they can be converted into crucial information sources in the core business. Therefore, the focus of this project is to explore this problem and try to give it a modular solution, adaptable to different realities, using recent technologies and one that allows users to access information where and whenever they wish.

In the first phase of this dissertation, bibliographic research, along with a review of the same sources, was carried out in order to realize which kind of solutions already exists and also to try to solve the remaining questions.

After this first work, a solution was developed, which is composed by four layers, and consists in getting the data to submit it to a treatment process (where eleven treatment functions are included to actually fulfill the multidimensional data model previously designed); and then an OLAP layer, which suits not just structured data but unstructured data as well, was constructed. In the end, it is possible to consult a set of four dashboards (available on a web application) based on more than twenty basic queries and that allows filtering data with a dynamic query.

For this case study, and as proof of concept, the company IOTech was used, a company that provides the data needed to accomplish this dissertation, and based on which five Key Performance Indicators were defined.

During this project two different methodologies were applied: Design Science Research, in the research field, and SCRUM, in the practical component.

Keywords: unstructured data, NoSQL databases, information retrieval, real-time analysis, OLAP.

INDEX

LIST OF TABLES

LIST OF IMAGES

ACRONYMS AND ABBREVIATIONS

A

ACID – Atomicity, Consistency, Isolation, Durability

API – Application Programming Interface

ADE – API Development Environment

B

BASE – Basic Availability, Soft state, Eventual consistency

BI – Business Intelligence

BSON – Binary JSON

BD – Big Data

C

CAP – Consistency, Availability, Partition Tolerance

CEP – Complex Event Processing

CRISP-DM – Cross Industry Standard Process for Data Mining

CSS – Cascading Style Sheets

D

DB – Database

DDD – Data-driven Decision

DI – Data Intelligence

DM – Data Mining

DS – Data Science

DSP – Distributed Stream Processing

DSR – Design Science Research

DW – Data Warehouse

E

EQL – Event Query Language

ETL – Extract, Transform and Load

F

FK – Foreign Key

G

GUI – Graphic User Interface

H

HOLAP – Hybrid Online Analytical Processing

HTML – Hyper-Text Markup Language

HTTP – Hypertext Transfer Protocol

I

IDE – Integrated Development Environment

IoT – Internet of Things

IP – Internet Protocol

IT – Information Technologies

J

JSON – JavaScript Object Notation

K

KDD – Knowledge Discovery in Databases

L

LAN – Local Area Network

M

MOLAP – Multidimensional On-Line Analytical Processing

N

NoSQL – Not only Structured Query Language

NPM – Node Package Manager

O

OLAP – On-Line Analytical Processing

P

PC – Pervasive Computing

PDS – Pervasive Data Science

PK – Primary Key

R

RDBMS – Relational Database Management System

RDF – Resource Description Framework

REST – Representational State Transfer

ROLAP – Relational On-Line Analytical Processing

S

SOW – Statements of Work

SK – Server Key

SQL - Structured Query Language

T

TCP – Transmission Control Protocol

W

WBS – Work Breakdown Structure

WORA – Write Once Run Anywhere

WWW – World Wide Web

X

XML – Extensible Markup Language

GLOSSARY

BSON

BSON, or Binary JSON, "is a binary-encoded serialization of JSON-like documents". This technology supports the inception of documents and arrays in other documents and arrays. Furthermore, it allows the representation of data types that JSON does not[1].

CSS

CSS, or Cascading Style Sheets, is a language used to describe the style associated with an HTML document, how its elements should be displayed [2].

FOREIGN KEY

A foreign key is at least one field in one table that is a primary key in another table. This key is used to connect to different tables becoming the original table the parent or referenced table and the one that receives the field(s) the child table[3].

HTML

HTML, or HyperText Markup Language, can be described as the structure of web pages that use markup. Furthermore, its elements are represented by tags that even though they are not displayed on the web site are used to render their content[4].

HTTP

HTTP, or Hypertext Transfer Protocol, is a group of rules for files transferring on WWW, World Wide Web. Also, it is an application protocol that is executed over TCP/IP ("suite of communication protocols used to interconnect network devices on the internet"[5])[6].

[1] http://bsonspec.org/
[2] https://www.w3schools.com/css/
[3] https://www.w3schools.com/sql/sql_foreignkey.asp
[4] https://www.w3schools.com/html/html_intro.asp
[5] https://searchnetworking.techtarget.com/definition/TCP-IP
[6] https://searchwindevelopment.techtarget.com/definition/HTTP

JSON

JSON, or JavaScript Object Notation, "is a lightweight data-interchange format". This technology is based on a subsection of the JavaScript programming language, and it is a text format that uses a set of conventions common to some programming languages. Beyond being easily understandable for humans is quite easy for machines to parse it and generate it[7].

NoSQL

NoSQL, or Not Only SQL, is a concept related to NoSQL databases, which are databases that are not based on SQL. In other words, NoSQL, refers t a set of databases that are not relational, have no schema and are intimately related to losing consistency models. These databases provide some kind of support for SQL-based interfaces even though they are not slightly relational [8].

PRIMARY KEY

Primary keys uniquely identify each record in a table and must contain unique values and cannot contain NULL values. One table can just have one PK, but it can be composed of different fields[9].

RESTful API

A RESTful API is an application that in order to get, put, post and delete data uses HTTP requests. This technology is based on REST technology (architecture for web services development recognized as simple and for being built over other systems and HTTP features[10])[11]

SQL

SQL, or Structured Query Language, is a language intended to deal with data storing, manipulating and retrieving in databases, typically relational databases[12].

[7] https://www.json.org/
[8] https://spring.io/understanding/NoSQL
[9] https://www.w3schools.com/sql/sql_primarykey.asp
[10] https://searchmicroservices.techtarget.com/definition/REST-representational-state-transfer
[11] https://searchmicroservices.techtarget.com/definition/RESTful-API
[12] https://www.w3schools.com/sql/

SURROGATE KEY

Surrogate keys are similar to primary keys in the sense that their goal is to uniquely identify records in databases structures. The difference is that it is the only utility in the system, while primary keys can have meaning in the context that they are inserted into. Usually, they are unique sequential numbers[13].

XML

XML, or eXtensible Markup Language, was developed with the main goal of store and transport data and in a way that it could be readable for humans and machines[14].

[13] https://www.techopedia.com/definition/22403/surrogate-key
[14] https://www.w3schools.com/xml/

CHAPTER 1 – INTRODUCTION

This first chapter aims at introducing this dissertation by giving it a context and motivation. Furthermore, the main goals, along with some structuring ones, are presented, making clearer what this project is about. Then the present document structure is presented to provide some guidance over it.

1. CONTEXT AND MOTIVATION

In the past few years, the constant growth of data volumes is notorious. That has become even more intense with the implementation of the Internet of Things (IoT) concept, described as a model that allows different devices (the "things") to be connected as part of the internet (Simmhan & Perera, 2016). These "things" are used to capture several kinds of data related to the environment they are in or related to human beings that use them. So, as one can guess, tons of data are generated by them and tend to increase according to the number of new devices that are integrated into this network. In 2018, around 23.14 billion devices were part of this IoT universe worldwide, and it is estimated that in 2025 this value might triplicate, reaching 75.44 billion (Columbus, 2016).

Furthermore, it is easy to understand that companies themselves have a lot of data surrounding them and need to process it in order to retrieve relevant information from it. Some of that data have the potential to bring real important insights to those companies' business decisions, but the trick is in the hidden patterns that are actually meaningful to them (Hurwitz, Nugent, Halper, & Kaufman, 2013).

Since the matter is sources of data with large data volumes, a wide volume variety and with a high data speed, it is plausible to call here the Big Data concept (Hurwitz, Nugent, Halper, & Kaufman, 2013). In an easy way, it can be described as a set of data sources that are too big in such a way that they demand different technologies to process them (Provost & Fawcett, 2013).

With this new level of complexity there are different technological needs, nevertheless, Big data "is a combination of the last 50 years of technology evolution" (Hurwitz, Nugent, Halper, & Kaufman, 2013).

Even though the SQL databases are profoundly ingrained in nowadays enterprises and systems, Big data brought NoSQL out onto the sunlight. This happens because they possess

mechanisms to handle large volumes of data and provide support for Big Data (Hashem, et al., 2014). More than supporting these enormous data volumes, other motivations are related to the simplicity of its design, its scalability and the fact that they allow similar queries as SQL language ones (Sareen & Kumar, 2015).

The fact that NoSQL databases are schema-free, or schemaless, makes them an excellent choice when one needs to store and manage unstructured data or non-fixed content, once Relational Database Management System cannot support other than fixed structured data, and even though it is possible to work on this by managing the database tables, when they reach certain sizes it becomes really unbearable in what concerns the processing time (Mitreva & Kaloyanova, 2013).

Well, Big Data systems' users want as less time as possible between data reception and the moment they can consult the process's results, and that brings another critical point to this thematic. In some cases, as understandable, it is extremely important for enterprises to have real-time data, so they can make updated business decisions and maximize the outcomes of the informed decision. That's another reason why NoSQL databases have become more used in real-time web applications (Kotecha & Joshiyara, 2017).

Since the problematic here presented lays in collecting data, working over it and displaying the results, it is correct to index it into the Data Science field (Davies & Clinch, 2017).

To sum up, there are some technologies to deal with huge data volumes, some other to support unstructured data and take care of real-time operations. But what if we mix all these topics and try to execute queries over unstructured data in useful time? Which techniques and technologies does one need to apply to solve this question?

Based on such questions, this dissertation has as final output a web solution capable of dealing with all sorts of data (structured, semi-structured and unstructured data), of retrieving information from them and displaying them in a comprehensive way to final users. In order to achieve that solution, two different layers were built: an API layer and then, logically, a web layer. As proof of concept data provided by the enterprise IOTech was used, more specifically from its application ioHub, that aims to make the process in the society of services simpler, by simplifying the communication between citizens and service providers.

IOTech is an enterprise whose main goal is "developing intelligent and innovative solutions able to simplify people's life and connecting them to an interactive world"[15].

2. GOALS

The main research question that lays on this dissertation's foundations is "How can we process such amount of different kinds of data in order to extract relevant knowledge in useful time?".

In order to answer this question, this project's main objective is the development of a prototype able to support consulting relevant information in useful time, regardless of the type of data used to create information beyond this interface. As proof of concept the company IOTech, which provided the data to test the output of this dissertation, was used. This data can only be used in this project's context, for its reproduction is forbidden.

The following are considered as main goals for this project:

[1] Development of a prototype to consult information through different Dashboards in useful time over an OLAP (On-Line Analytical Processing) structure;

[2] Development of an API, Application Programming Interface, to get data, prepare it and store it;

To support them, a few structuring goals were defined for each one, as follows. For the first main goal presented:

[1.1] Develop a Multidimensional Model.

[1.2] Develop an OLAP layer.

[1.3] Develop a Web Paige to support the Dashboards.

[1.4] Develop a set of Dashboards.

Regarding the second main goal:

[2.1] Analyze and Select Data.

[2.2] Treat the Data to be used.

[2.3] Develop mechanisms to handle Unstructured Data.

[2.4] Develop mechanisms to retrieve Information.

[15] https://iotech.pt/

So, at the end of this dissertation, as final result, a functional web prototype that suits its purpose with a pervasive interface, and which improves the analytics data process in useful time, was obtained. Furthermore, the knowledge of information systems, web programming, and data science was extremely increased.

To achieve those objectives, a literature review in all the fields needed to approach this theme properly was necessary. Research is fundamental, in order to know what type of possible solutions to the problem already exists, once it allows knowing what can be introduced as new and what can be improved.

3. STRUCTURE OF THE DOCUMENT

The present document is designed according to a book's structure, therefore, is divided into 5 chapters with a proper introduction, conclusion and notation. It includes the following chapters:

- Introduction: the abstract, context and motivations for this theme are exposed, as well as its goals and expected results;
- Literature Review: in this chapter the result of the research in the kind of existing proposals to this problem is presented. It allows to define some important concepts related to the dissertation's theme as well as what this on-going dissertation can improve or add as new to the field being studied. This chapter is divided into five main sections: Context, Data Science, Big Data, NoSQL Databases, and Data Analysis;
- Methodological Approach: the methodologies selected to suit this project (SCRUM and DSR) and a brief description of each one are presented;
- Practical Work: the work developed through this project is reported, by presenting all methods and materials used, a solution plan, along with its underlying development layers, and the case study used as proof of concept;
- Project Plan: here it is possible to find information associated with the project's management, such as tasks, deadlines and the project's risks;
- Conclusion: completes the document and it is also used to present what the following work might be;
- References: contains a list of the bibliographic references used along with the present document.

CHAPTER 2 – LITERATURE REVIEW

This chapter introduces several concepts relevant to this dissertation's development and it presents existing works on the field, and its actual situation. Thus, the dissertation's theme can be presented along with its state of the art.

1. INTRODUCTION

This phase takes place between November 2018 and January 2019 and started with a comprehensive search that turned out being fractioned into several pieces (according to thematic groups) and just then grouped in this document. This is clearly exposed in this chapter's structure. First of all, there is an explanation on how the documents used to write this document were selected and just then the concepts are presented. In this second part it is possible to find five main sections: the first one (the present section) is used to introduce the literature review and briefly explain what strategy was adopted to do so; the second one the introduces the theme of Data Science and briefly explores what Pervasive Computing is, also redirecting to its subfield, Pervasive Data Science; the third section approaches the theme of Big Data and its problematics; the fourth section is related to NoSQL databases as unstructured data supporters; the fifth and last section goes through the topic of Data Intelligence, approaching data and business intelligence, giving particular attention to OLAP layer and how it must be handled considering non-structured data, real-time data and data analysis and, in the end, some solutions are proposed.

To accomplish this, and minimize the waste of work and time, this phase started by defining some "rules" to restrict which documents might really be relevant to this work (of course, in some particular cases, there might have been exceptions). This strategy goes through:

- The use of different indexing services such as Scopus, Web of Science, ScienceDirect, IEEE, ResearchGate and Google Scholar;
- The search could just be in English or Portuguese;
- The documents' publication year must be from 2010 ahead, except in a few cases of documents relevant in the study field;
- The documents must be journal articles, papers, books or other dissertations;
- First, reading the abstract and conclusions and then evaluating the contents' relevance.

This research is based on concepts such as:

- Services Society;
- Pervasive Data Science;
- Unstructured Data;
- Semi-structured Data;
- OLAP;
- Construct Structured Data;
- Extract Knowledge;
- NoSQL and SQL Data Warehouses;
- Real-time Data Analysis.

2. SOCIETY OF SERVICES

First, it is important to understand what a service is. According to Steve Jones (2005), a service can be described as being a "discreet domain of control that contains a collection of tasks to achieve related goals", in other words, a service is contextualized in a specific environment and involves performing several tasks in order to achieve specific goals related to its context. Services can be performed either by humans or machines according to the service needs (Meirelles, 2006).

According to the Portuguese Civil Code[16], article number 1154, provide service means to celebrate a contract in which one of the parts involved compromise himself into providing a certain result of his intellectual or manual work with or without retribution.

The range of what can be considered as a service variates between a customer helpdesk until the using of sophisticated tools to provide companies assistance in the most variated areas.

The use of technology is continuously increasing, and it has been applied to the most variated actuation fields once its efficiency has been proved to help to solve those field problems. Well, services are no different, and when this fact is crossed with the presented ideal of what is provide a service, the terminology Services Society was born.

A Services Society is a set of services that are provided to the society in favour of it and its citizens being this connection made using technologies (applications, and others).

[16] https://www.igac.gov.pt/documents/20178/358682/C%C3%B3digo+Civil.pdf/2e6b36d8-876b-433c-88c1-5b066aa93991

The outcome of this project has as a final goal provides to its users a set of relevant dashboards displaying information in useful time. In this way, this dissertation theme, "Pervasive Data Science applied to the Services Society", can be contextualized.

3. DATA SCIENCE

In this section, it is presented the Data Science (DS) concept and how it relates to the past topics. This subchapter must start with a proper introduction of what is Data Science after so, according to Cady (2016), "Data science means doing analytics work that, for one reason or another, requires a substantial amount of software engineering skills", in other words, it "is a set of fundamental principles that support and guide the principled extraction of information and knowledge from data" (Provost & Fawcett, 2013).

To help to understand what actually data science is, and what is it made about, Figure 1 is presented. On in it becomes clear that data science can be expressed as the field that involves analysis, particularly data analysis, structure, a set of algorithms to extract knowledge, a data process phase, requires programming skills to solve its questions, always with the main goal of solving real problems by reaching knowledge to support this solving process.

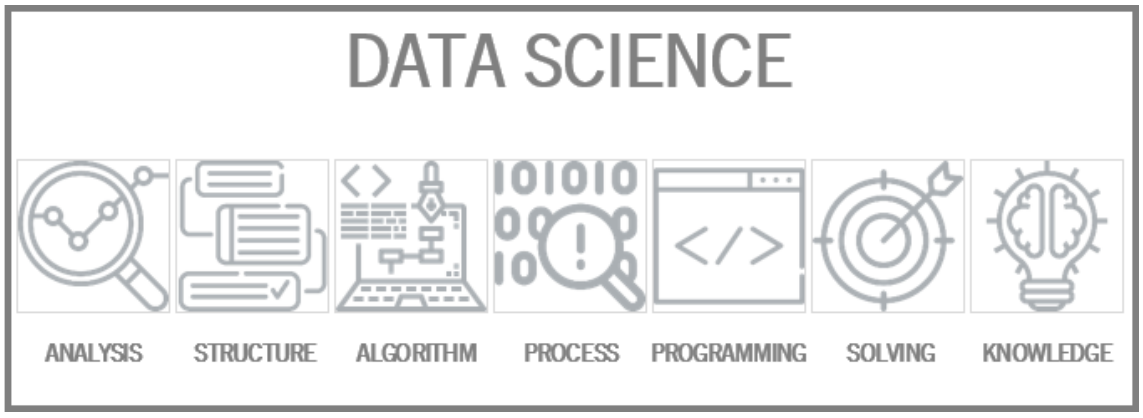


Figure 1 - Data Science.

3.1. DATA SCIENCE ROAD MAP

Defined DS, it is inevitable to take a look into the road map proposed by Cady (2016) that is possible to see in Figure 2. This as the purpose to present, in a high-level way, the path taken to solve a data science problem.

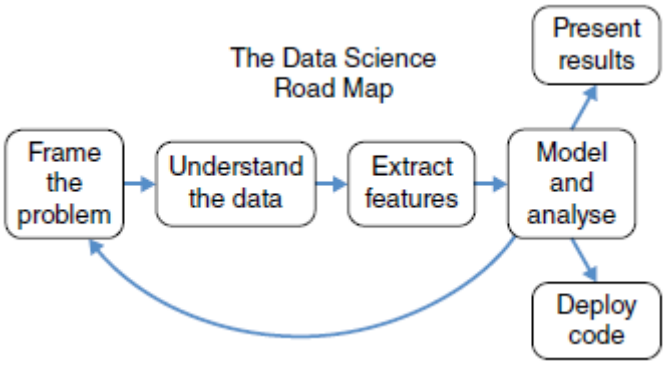


Figure 2 - The Data Science Road Map (adapted from (Cady, 2016)).

So, the first step is "Frame the Problem". This step consists in understanding what is really needed in order to redirect the work to produce a solution that actually will solve the client problem. Here it is important to ask the right questions, so, knowing which criteria constitute a completed project and what would be mandatory to the project achieves success becomes vital. One way to do so is by inditing an SOW (Statements of Work) while discussing the terms with the stakeholders. This document helps to maintain a compromise from both parts but, of course, might be submitted to reviews if necessary.

"Concluded" this first step, appears the "Understand the Data" phase. Here is the time to analyze the data that are submitted to work. To do so, this phase can be split into three subphases:

- Basic Questions: this is the first thing to do to understand the available data, ask basic questions related to it. Some of them related to the physical form (size, structure), others related to the comprehensiveness (good representation, part/full dataset), others with data quality (errors that might be found, blanks presence) or even related to the need of joining different datasets and with its implications (SOWs might include a description of the available data). Albeit all these questions are important, there is one that cannot be missed: "Does this data solve the problem in hands?", if not, other sources might be needed and the work plan change;
- Data Wrangling: in this step is time to transform raw data into something data can be used as input to conventional analytics. Basically, the data are submitted to filtration and cleanup. Here is where it is required more DS skills, so it would be possible to deal with complex pipelines and messy data;

- Exploratory Analysis: this phase has as purpose transform data, so it is possible to see it from different perspectives and deepen the data understanding. Here it is possible to go beyond the existent data and calculate some correlations so it can be seen in other ways. From this phase can come two different outputs: an innate data perception with a visual insight of data patterns or a hypotheses list motivated by previously generated graphics.

After this important step, comes the "Extract Features" (feature can be a number or category, extracted from the data, that describes an entity). This one can be simultaneous with Data Wrangling and Exploratory Analysis. Essentially, this consists of getting raw data into tabular data (where rows are real-world entities and columns gives a single piece of information associated with a specific entity), data that analytical techniques can work upon. Getting good features is what makes analysis work. In order to get that, the data scientist must get close to domain experts, so they get to know does real-world phenomena means and get it into numbers the best way possible. The majority of these features might be used to predict something, the target variable, which must be extracted as well.

Finished features extraction, starts another phase "Model and Analyze". This step is related to the conception of a machine learning model, which complexity variates according to the scenario in which it is applied. Furthermore, it is here that is decided if it is needed another iteration and what to change on it.

After "Model and Analyze", according to the client type and the process needs, there are three possible ways:

- Present Results: this one applies for cases where the final client is a human, and often if it is a machine as well and consists of describing the work done and ultimate results. Here becomes important the ability to communicate with people from different expertise areas about technical content in a way they could understand;
- Deploy Code: this one applies only if the client is a machine, and allows taking two different paths, Batch Analytics Code (which can produce reports or train models that will be used by other code) or Real-Time Code (usually, consists into an analytical model). In the end, must be outputted the code, a run book and unit tests (for real-time code) or an input sample along with the expected result (for batch processes);
- Iteration: this might or not happen according to the success of the previous iteration. In order to improve the work quality, the author advises getting sooner results in order

to redirect the analysis, automate the analysis in just one script and maintain the code modular to simplify future changes.

3.2. CRITICAL CONCEPTS

Now that it is possible to see, in a high-level description, what the Design Science is made off some of its critical concepts, according to Provost and Fawcett (2013), are presented in order to deepen the lore on this topic:

- Methodologies, like CRISP-DM, help to find a structure in the process of knowledge extraction from data when the idea is to solve a business problem;
- The results of the data science process must be evaluated taking into consideration the environment they are into;
- Between the business problems and analytics solutions subproblems can be found and "attacked" separately and just later combined;
- Data Science allows to find correlations inside the data available to find out more information about the data itself;
- Entities similar considering known features are likely to be similar considering unknown ones;
- "Overfitting" is something to be careful with once it might implicate the data mining (DM) model could not produce accurate results when applied to different data sets;
- All methods applied in order to have some conclusions involve some assumptions but do not accept confounding factors.

3.3. DATA-DRIVEN DECISION MAKING

But what is the purpose of executing all work described in the last topic? According to Provost and Fawcett (2013), the main goal of DS is improving the decision-making process, rendering to the business interests.

So, as the designation suggests, data-driven decision (DDD) is about making decisions based on data analysis instead of something else (Brynjolfsson, Hitt, & Kim, 2011). The impact that DDD has on companies has been confirmed in a study developed by Brynjolfsson, Hitt, and Kim. They have shown that productivity increases, paired with how much the company is data-

driven. This comes to enhance the importance of investing in Data Science structures. Sometimes, these two concepts (DDD and DS) can be overlapped when the machine can decide something based on the DS process. This can be seen clearly looking at Figure 3.

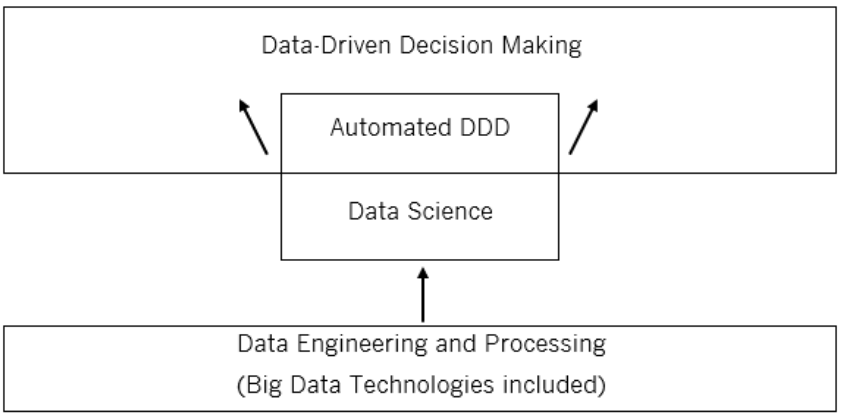


Figure 3 - Data Science relationship with DDD (adapted from Provost and Fawcett (2013)).

In this scheme, it also becomes clear that not all data processing is included in what is data science. For instance, data engineering and data processing have shown, support the all DS process and consequently the DDD, but that does not make them part of DS. This happens because they are useful not only on this topic but in many more others. Here appears the Big Data concept, which is explored in the next subchapter.

3.4. PERVASIVE DATA SCIENCE

Now that Data Science was properly introduced in this document, it is needed to understand what Pervasive Data Science is and what does it involve. In order to do that, first, it is also necessary to understand the basic ideas of what pervasive computing is.

So, according to Kurkovsky (2008), Pervasive Computing can be defined as the idea that "computing equipment will grow smaller and gain more power; this would allow small devices to be ubiquitously and invisibly embedded in the everyday human surroundings and therefore provide easy and omnipresent access to a computing environment". Understanding this ideal makes it easy to recognize the main challenges associated with this research field as minimizing the impact that these systems might have on user's perception and how to make a system invisibly built-in the environment.

An easy and simple way to frame Pervasive DS would be describing it as an intersection between two other study fields, Data Science and Pervasive Computing. Starting from this ideal makes sense that Pervasive Data Science (PDS) pursuits the main goals from both fields, which turns the main goal of PDS collect and analyze data building knowledge while chasing the invisibility of this system in the environment (Davies & Clinch, 2017).

By association, the challenges are also a mix of both areas. Keeping in mind that data science main challenges are related directly with Big Data Vs that are introduced in the next subchapter (velocity, volume, variety and veracity) and that development of appropriate systems architectures, different interactions with users, etc. are problems faced by pervasive computing (PC) it is natural that some of the challenges of PDS might be (Davies & Clinch, 2017):

- Management of data ownership complex models;
- Ensure data in pervasive systems;
- Balance data demand whit privacy concerns;
- Impact of PC in DS challenges;
- PDS environments architecture;
- Access rich data through pervasive technology;
- New data-driven actuation forms.

Fundamentally, here data science knowledge and the pervasive computing ideals are melt together, by other means, invisible and omnipresent data science models.

To understand a little bit more the concept of pervasiveness it is presented the Figure 4, which represents the “Pervasive Information Systems Value Chain” according to Angelov and Rao (2008). For them, it is composed of four layers: Infostructure, which is the information systems infrastructures, Devices, Interfaces, and Smart Spaces; and in each layer, a new value is created.

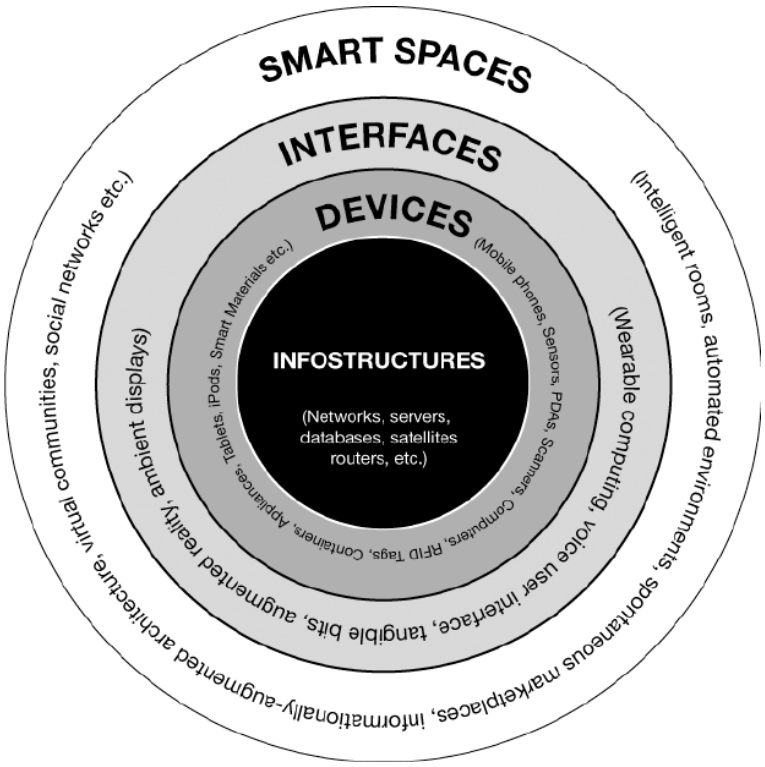


Figure 4 - Pervasive Information Systems Value Chain (adapted from Angelov & Rao, 2008).

The Infostructures layer is the one the includes "communication channels, data and sensor networks, and broadband internet access, but also the intelligent design of databases, storage, and network applications".

Regarding the Devices layer, it is safe to say that it is a layer that is intimately related to the end-users service experience or with the proper device experience. Devices appear here to represent all members of a pervasive system that is not a human being, such as mobile phones, sensors, vending machines, ATMs, and so on. Keeping in mind that the idea beyond pervasive computing involves the existence of many devices surrounding the user, these devices have to work with each other along the way in order to provide a unified service experience.

The upper level is Interfaces, which is related to interaction design. Here pervasive applications face a severe challenge which is the development of interfaces that are integrated into the environment where they are put into by being intuitive and modular. It is fair to prospect that pervasiveness is actually experienced through these interfaces, once they are the responsible for "for translating the complexity of the computational environment and service offering into a more useful and accessible to the end-user".

Finally, the top-level is Smart Spaces, which is related to the idea of an interaction that has some value between devices and the environment where they are inserted and for that, space,

where they exist, has intelligent properties. To the authors, smart spaces are related to providing services that have on their base intelligence acquired from the infostructure, which then is utilized by devices and accessed through interfaces.

Therefore, it is important to recognize that all the presented levels are integrated somehow, pursuing ubiquity and omnipresence and for that, the value chain, even divided into layers, must be seen as a whole.

4. BIG DATA

Nothing more appropriate than start this chapter by introducing Big Data. This term has been growing these past few years since we are living in an era where data are in the centre of our daily activities, either in companies' context or personal lives (Zikopoulos, Eaton, Deroos, Seutsch, & Lapis, 2012). So, what is Big Data after all? What characterizes it?

By Hurwitz et al. (2013) definition, Big Data is "the capability to manage a huge volume of disparate data, at the right speed, and within the right time frame to allow real-time analysis and reaction".

For Krishnan (2013), Big Data is defined "as volumes of data available in varying degrees of complexity, generated at different velocities and varying degrees of ambiguity, that cannot be processed using traditional technologies, processing methods, algorithms, or any commercial off-the-shelf solutions".

Well, according to Zikopoulos et al. (2012), Big Data is the term applied to information that can't be processed or analyzed by traditional methods and, like the Krishnan, affirms that it is defined by three characteristics: volume, variety, and velocity, the V^3. Figure 5 can clarify it.

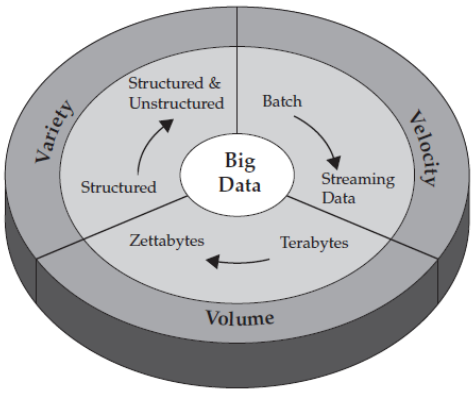


Figure 5 - Volume, velocity, and variety IBM characterization (Zikopoulos, Eaton, Deroos, Seutsch, & Lapis, 2012).

But how do we know if we are facing Big Data? Which are the boundaries? Well, there is not a well-defined limit for what is or not big data in numbers matter, it depends on the context and of the time window considered. For instance, it is possible to consider a huge volume of data, even if it is simple data, as big data but also a relatively small set of data that is very dissimilar and complex.

That's why, for Hurwitz et al. (2013), reducing Big Data (BD) to this three Vs, even it might be convenient, can also induce an illusion and become extremely simplified when compared to the reality. Thereat he refers to the importance of a fourth V standing for veracity.

In a simple way, it is plausible to refer to volume as the quantity of data, velocity as the data process time and variety as the different types of data involved in this process (Zikopoulos, Eaton, Deroos, Seutsch, & Lapis, 2012). As for veracity, it is considered the accuracy of the data, is the sense of the result of big data analysis (Hurwitz, Nugent, Halper, & Kaufman, 2013).

4.1. VOLUME

So, data volume, as referred previously, is related to the amount of data generated continuously. In the past few years, the data volumes have been increasing in a way that now when the talk is about big data it already include values in zettabytes It is important to keep in mind that different types of data come with different size and therefore the data set have different volume considering it (Krishnan, 2013).

4.2. VELOCITY

Regarding velocity, it is imperative to distinguish the batch process from data streaming and its relevance for the case. Previously it was used to analyze data in, typically fixed-size, groups adding delay into this process for what the purpose was using this latter than the moment it was received (Krishnan, 2013). Once in the BD era, it became mandatory being able to deal with streaming data, applying the process in real-time or real close to it. So, to be effective, BD must execute data analysis against huge volumes and a variety of data while it is still receiving new data not just when it is "paused" (Hurwitz, Nugent, Halper, & Kaufman, 2013).

4.3. VARIETY

Another question raised by BD was the variety of data. First of all, it needed to understand which kind of that exists and what distinguishes them. Thus, here it is presented the three kinds of data that are possible to find:

- Structured Data, the most common and studied type of data. This is the data that can be included in relational databases for being highly organized and structured, "An instance of such a schema is some data conforms to these specifications" (Sint, Schaffert, Stroka, & Ferstl, 2009). This allows for efficient search (Kanimozhi & Venkatesan, 2015);
- Semi-structured Data, this can be seen as one particular type of structured data that is deprived of a well-defined structure. This means that this kind of data does not need a schema definition, it can have markers that separate semantic elements and enforce hierarchies. So, it can be described as "schemaless or self-describing, terms that indicate there is no separate description of the type of structure of the data" (Sint, Schaffert, Stroka, & Ferstl, 2009). JSON and XML are examples of languages used to manage this type of data (Kanimozhi & Venkatesan, 2015);
- Unstructured data, this one is the most complex type of data and can be divided into two types: Non-Textual, like images, videos and audio files, and Textual unstructured data, like emails, messages, collaborative software, word documents, PowerPoint presentations, etc. (Kanimozhi & Venkatesan, 2015). The unstructured concept remit to the idea of no recognized structure for the present data (Sint, Schaffert, Stroka, & Ferstl, 2009).

This variety increases the problem complexity since the data processing is depending on the availability of proper metadata to inform which kind of information is really there. This adds pre-requisites for the process, like scalability, distributed, image processing, graph, video and audio capabilities (Krishnan, 2013).

4.4. MORE CHARACTERISTICS

After this brief explanation became easy to understand why Big Data is so complex and which kind of problems it goes against. That's why Krishnan (2013), mention another three characteristics: Ambiguity, Viscosity, and Virality. These concepts take place in the Vs "edges", in other words, ambiguity, which is the uncertainty caused by the lack of metadata are a problem allocated between volume and variety once it crosses both themes, viscosity is placed between volume and velocity thematic once it is related to the resistance to flow in the volume data, and virality takes place amid variety and velocity since it measures the speed with the data are spread in a people-to-people network (Krishnan, 2013). This is clarified with the following image (Figure 6).

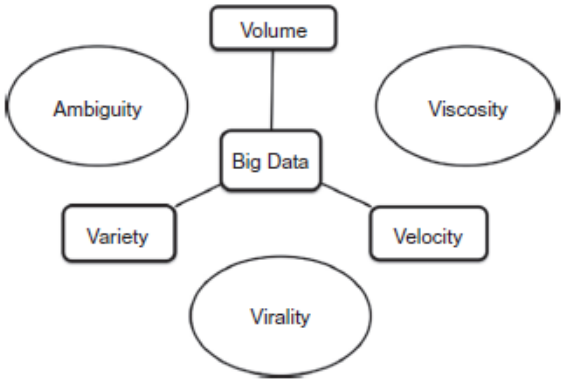


Figure 6 - Other Big Data Characteristics (adapted from Krishnan (2013)).

5. NoSQL DATABASES

After introducing Big Data and some of its problematics becomes easier to understand the role that NoSQL databases have in this thematic. Before going deeper into this vast subject, it is vital to understand what it is and how distinguishes itself from SQL databases.

5.1. RELATIONAL DATABASES VS NoSQL DATABASES

For many years Relational Database Management System (RDBMS) has been the first choice to store data since it is well structured with columns representing a specific type of data and rows representing an entry in the database (DB), both forming data tables that must always

have a primary key. A SQL database, or relational DB, is composed at least for two tables which must be connected, forming the DB schema (Kotecha & Joshiyara, 2017). In these databases it is possible to perform focused queries obtaining very formatted results which is a vantage (Kanimozhi & Venkatesan, 2015), but they have the same limitation to take in consideration: Scalability, to do so it is required powerful servers that bring high costs and complexity to organizations, and Complexity, their rigid structure forces data to fit into tables, if not, a more complex DB design is required adding difficulties into DB management (Kotecha & Joshiyara, 2017). Thus, despite SQL has been a reference in databases context for several decades resulting in tons of documentation and in high level of maturity (Hammes, Medero, & Mitchell, 2014), now, facing new conditions, as the ones that Big Data implies, it is time to search for more suitable databases technologies.

In Figure 7, it is possible to look over RDBMS structure example (where PK is used to indicate the column correspondent to the primary key)

ID (PK)	Name	Number	University	City
1	Ana	00001	UMinho	Braga
2	Rui	00002	FEUP	NULL
3	Camilo	00003	NULL	Coimbra

Figure 7 - RDBMS Structure Example.

To work around these problems, around 2009, appeared NoSQL databases. Well, NoSQL databases, as the name implies, are called non-SQL DBs or non-relational DBs, more consistently, Not-Only-SQL DBs, (Sareen & Kumar, 2015). According to Kotecha et al. (2017) and Sareen et al. (2015), NoSQL databases can be defined as databases that allow to retrieve and store data that is not modelled rendering tabular relations used in relational databases.

In Figure 8, compare its structure to an example of a NoSQL structure, in this case, Document Stores, and how the elements take place in both databases.

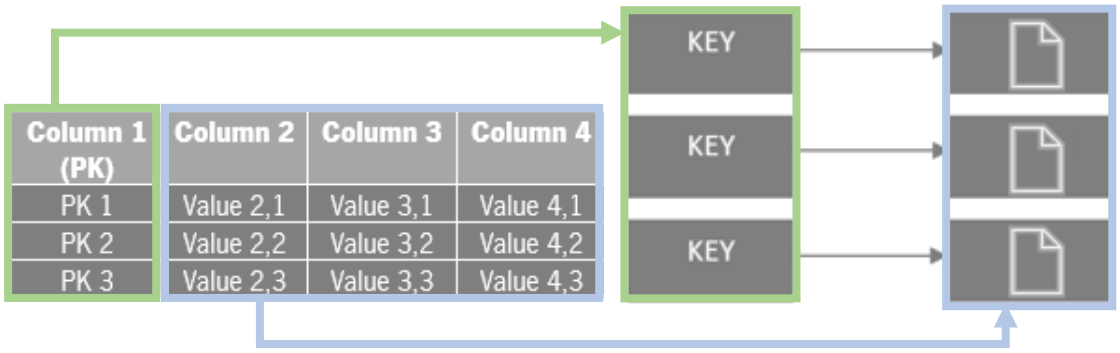


Figure 8 - Comparison between RDBMS and NoSQL (Document Stores in the case).

In Figure 8, was intended to show that RDBMS and NoSQL structures are away different, but there is also possible to see some inside structures that behave similarly. So, the primary keys in RDBMS have quite the same functions as the keys in NoSQL document stores once in both cases they must be unique with the aim of identifying an entry in the first case and a document in the second one. Furthermore, the data contained in the values disposed over rows in the first case, would be disposed of over documents in the second one. This thinking can be extended over NoSQL key-value databases, where the key can be thoughted of as similar to a primary key in RDBMS and the value to the remaining row content.

5.2. NoSQL DATABASES CHARACTERISTICS

Even though NoSQL databases are quite different from SQL ones, some of them as grown because allow the user to "mix both worlds", this means that yet they are using NoSQL DBs, some of them are based on entries and provide SQL functions such as sorting, indexing, projecting and querying that must be used carefully considering the data models' complexity (Hammes, Medero, & Mitchell, 2014).

Regarding NoSQL, it is imperative to mention its main characteristics in order to deepen the comprehension of it.

5.2.1. BASE

First, it is presented in the BASE paradigm. Contrary to SQL databases, NoSQL DBs do not use ACID properties – Atomicity, all work in a transaction must be completed Consistency, the

database can change but both, start and end, states must be consistent, Isolation, the changes in one transaction cannot be seen till that transaction is finished, and Durability, the results after concluded transactions must survive to failures – but BASE – Basic Availability, which means that data must be available even it is consistent or not, Soft state, the system state might change without any input, and Eventual consistency, data does not need to be always consistent, but at some time it will be – which is less strict than the first one (Mitreva & Kaloyanova, 2013).

5.2.2. CAP

Then, CAP Theorem appears as an important topic to be discussed. So, CAP involves Consistency (all endpoints must have the same view over the data), Availability (all users can access the data whenever they want) and Partition Tolerance (the system keeps running independently the physical network partitions) (Padhy, Patra, & Satapathy, 2011). The problem here is that no matter what, existing databases must choose only two of the three CAP characteristics (Hammes, Medero, & Mitchell, 2014). RDBMS DBs so far take ahead consistency and availability, but in order to offer partition tolerance, NoSQL DBs gave up one of the other two, having in preference availability taking, as mentioned, eventual consistency as "enough" (Burd, 2011).

5.2.3. MAP-REDUCE

One of the central notions associated with NoSQL databases is Map-Reduce. This "is a framework the supports the handling of large data volumes over distributed network nodes" (Mitreva & Kaloyanova, 2013). Since one of the problematics of the topic in hand is the huge volumes of data, it becomes useful to split this data and subsequent problems to work on effortlessly and address the work in portions to the different network nodes. This is the map phase. After the problems are solved, these nodes do the reverse process and send all the results to the master node, so it can interpret them. This is the reduce phase (Scholz, 2011).

5.2.4. MORE CHARACTERISTICS

There are other characteristics that are important as flexibility, redundancy, no declarative query language, etc. but scalability is the last characteristic that is loomed in this document. NoSQL

DBs allow to horizontally scale operations into many servers and split the data over them as well. To apply queries over the results, they must come from the same server. Otherwise, it will take more time to get an answer losing all scalability profits (Mitreva & Kaloyanova, 2013).

After this, it is possible to canvass some benefits of using this type of database. At the top of the list should be its high and easy scalability, then the fact that is less expensive to maintain NoSQL servers than SQL ones, redundancy (in order to avoid information loss), the fact that they can handle all data types, easier manageability and support integrated caching (Kotecha & Joshiyara, 2017).

5.3. LIMITATIONS

Like all other technologies, nothing is perfect, so there is possible to find also a few limitations. These ones are related with the disparity between NoSQL DBs implying the non-existence of standards, not availability of GUI (Graphic Users Interface) tools to manage these databases and the fact that this technology is so young that turns the mission of finding an expert impossible because of developers still in learning mode (Kotecha & Joshiyara, 2017).

5.4. NoSQL DATABASES TYPES

Now that a basic introduction to NoSQL is presented, it is time to present which kinds of NoSQL databases exist nowadays, according to Cattell (2011). Nowadays, there are already more than 225 NoSQL databases[17] and they can be divided into the following four categories.

5.4.1. KEY-VALUE STORES

These are the simplest NoSQL databases and works based on hash tables – "data structure that maps hashable keys to values. It does that by hashing the keys into the range [0, N] and using the hash as an index in a length-N array" (Cady, 2016). On those, the access to data is made through strings, called keys, and the format of data is not mandatory (Kotecha & Joshiyara, 2017). Some examples of this kind of DBs are Redis and Riak.

[17] http://nosql-database.org/

In the following images, the base structure of this type of database (Figure 9) and an example of data storage according to it (Figure 10Figure 10).

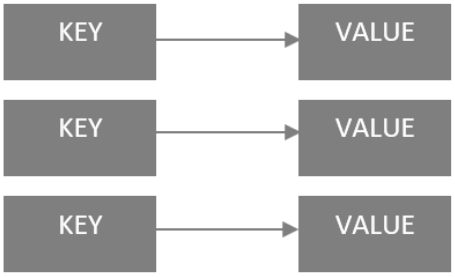


Figure 9 - Key-Value Database Structure.

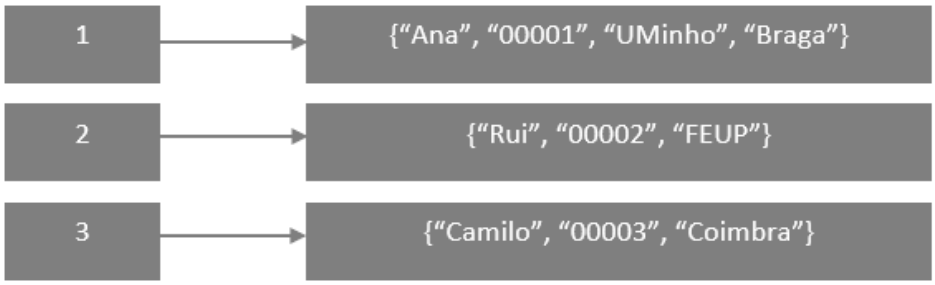


Figure 10 - Key-Value Database Structure Example.

5.4.2. DOCUMENT STORES

This is a “more complex version” of Key-Value databases and distinguishes the first ones by the fact that these have almost limitless capacity to save more instances. Here each document has its own data and key. They are used to retrieve, store and manage data and still slightly structured (Kotecha & Joshiyara, 2017). In document-oriented databases, the data are stored in JSON, BSON, XML format (Mitreva & Kaloyanova, 2013). MongoDB is the most important DB from this type, but there are others such as CouchDB.

In the following images, the base structure of this type of database (Figure 11) and an example of data storage according to it (Figure 12).

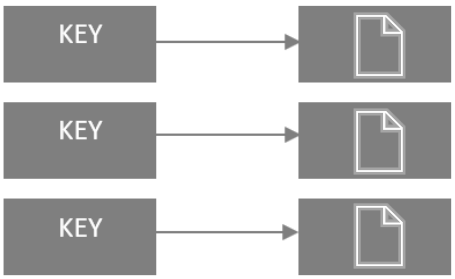


Figure 11 - Document Stores Structure.

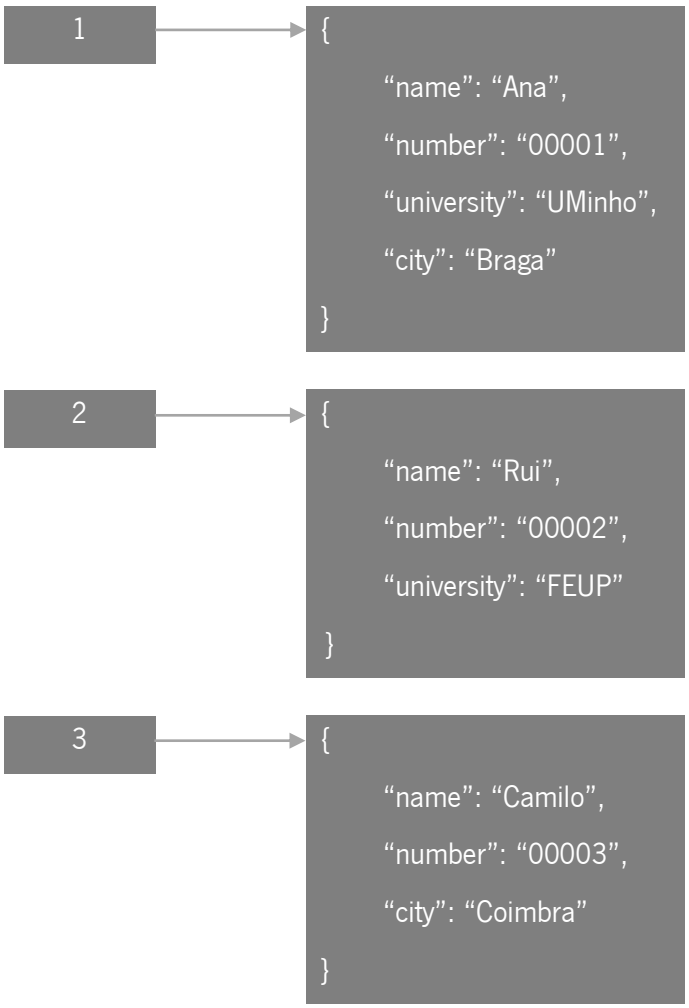


Figure 12 - Document Stores Structure Example.

5.4.3. COLUMN STORES

In this type, the column is the smallest instance of data and is nothing more than a tuple that contains a name, a value, and a timestamp. This format increases scalability and performance (Kotecha & Joshiyara, 2017). In this case, HBase, C-Store, and BigTable are good examples.

In the following images, the base structure of this type of database (Figure 13) and an example of data storage according to it (Figure 14).

ID	Column 1	Column 2	Column 3	Column 4
1	Value 1,1	Value 2,1		Value 4,1
2		Value 2,2	Value 3,2	Value 4,2
3	Value 1,3	Value 2,3	Value 3,3	Value 4,3

Figure 13 - Column Stores Structure.

ID	Name	Number	University	City
1	Ana	00001	UMinho	Braga
2	Rui	00002	FEUP	
3	Camilo	00003		Coimbra

Figure 14 - Column Stores Structure Example.

5.4.4. GRAPH STORES

This type of database is more used to represent networks and might induce high levels of complexity (Sareen & Kumar, 2015). Graph-oriented databases implement ACID kind of transactions. The most used database from this type is Nwo4j (Mitreva & Kaloyanova, 2013).

In the following images, the base structure of this type of database (Figure 15) and an example of data storage according to it (Figure 16).

Regarding Figure 16, it can be read in the following way: there are three nodes, two of them are composed by university members (Ana and Camilo) and another one stores information regarding a department of the university. The first two nodes are connected to each other, being its relationship defined as “colleagues”. The first two are both connected to the third one being this relationship defined as “members”.

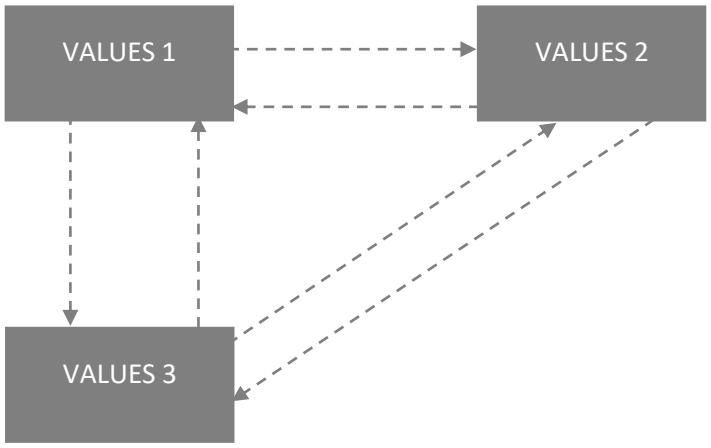


Figure 15 - Graph Stores Structure.

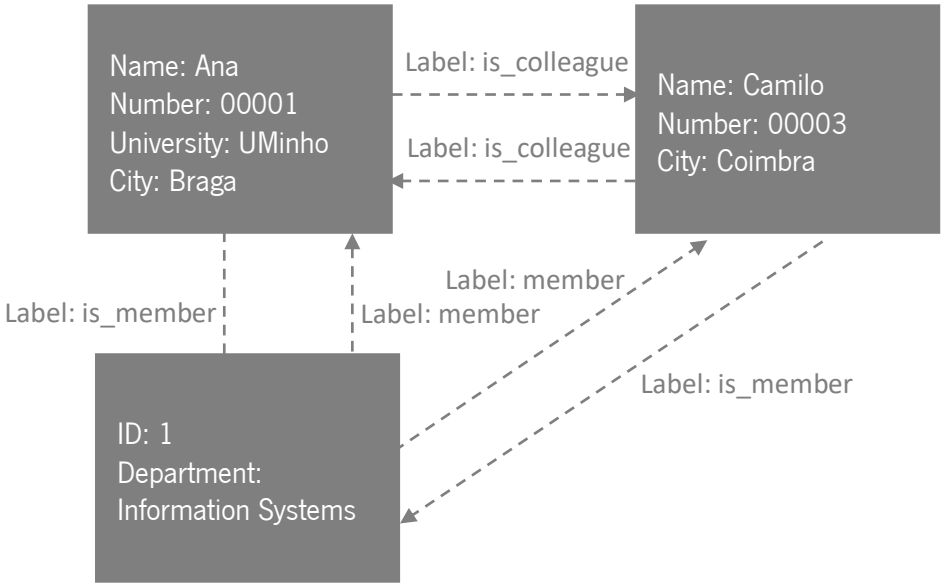


Figure 16 – Graphs Stores Structure Example.

6. DATA ANALYSIS

Even though enterprises have access to lots of data and this keeps growing, many times they are not effective in what concerns their usage. That happens because they do not spring the benefits of holding up this data and consequentially do not explore the value that it might have if they squeeze it into proper knowledge and business insights by processing and analyzing it or they are not capable to deal with large volumes of data or infer its quality (Füser & Georgi, 2013).

First, an enlighten on what is analytics after all. According to Bart Baesens (2014), "Analytics is a term that is often used interchangeably with data science, data mining, knowledge discovery, and others". All of those consist of extract relevant patterns or mathematical decision models. But why is it interesting?

Well, those models allow knowledge construction and consequence value creation for enterprises. Analytical models have been increasingly adopted to provide guidance in business most risky decisions (Baesens, 2014), once it was already proved (as mentioned before) that data-driven decisions are most likely to achieve success.

Since such responsibility lays on analytics "shoulders", and whatever the outcome is, will impact the organization, analytical models must be constructed under critical look so they will be optimal and lead to accurate information.

Regardless of this subchapter focusses on data analysis, some concepts as Business Intelligence architecture became important to refer to even just its "last phase" can be considered in this analytics field, and that is why On-Line Analytical Processing (OLAP) as an important role here.

More than that, along with this section, some proposed solutions for the knowledge extraction from unstructured data and the real-time analysis are presented when appropriate.

Further, in this subchapter, it is possible to find a comparison between the different tools for data analytics.

6.1. DATA INTELLIGENCE

As mentioned previously in this document data per se does not have proper value and interest, the results of its processing are the ones that really matter and bring value to organizations by allowing users to extract actual knowledge from there. Thus, it is plausible to refer Data

Intelligence in a simple way as the process that transforms raw data into value, in the last instance (Füser & Georgi, 2013).

With no more reprieves, Data Intelligence (DI) can be defined by a set of analysis through heterogeneous data (such as business performance, data mining, online analytics, and event processing) that have as main goal increase the informed decision power to their users (Rai, 2016).

In the words of Rohit Dubey (2018), DI has five components, each of them with a set of techniques associated. This can be clarified with the following schema (Figure 17):

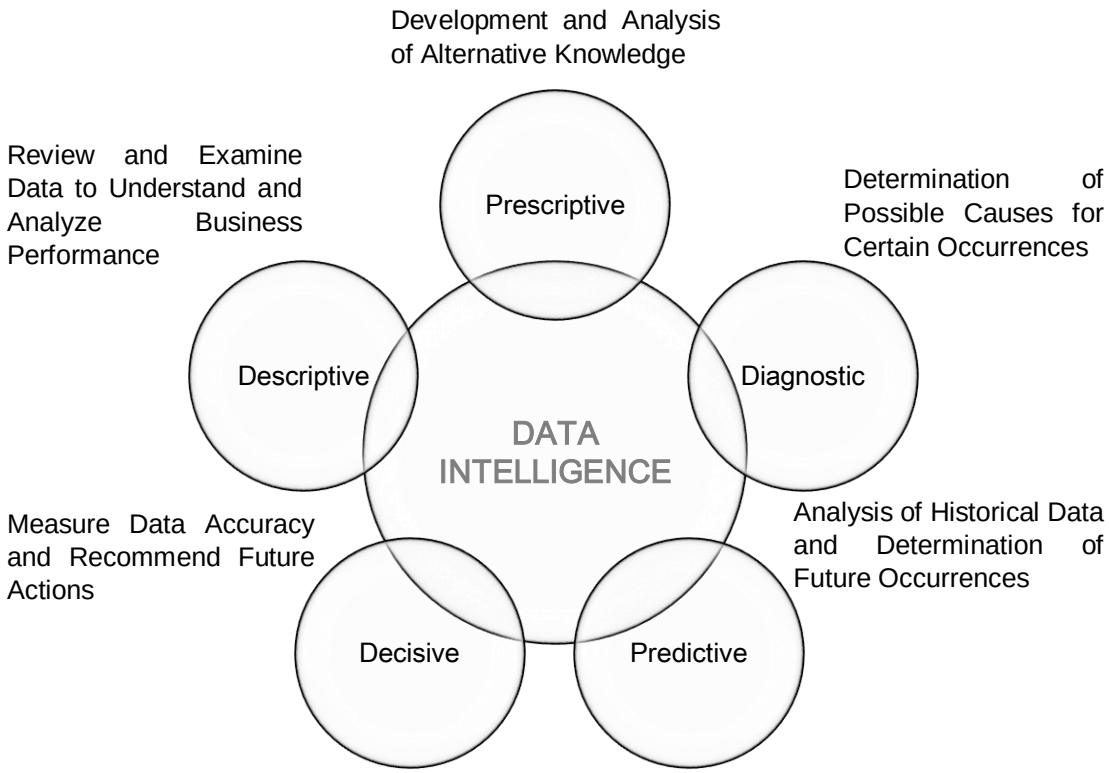


Figure 17 - Data Intelligence Components.

According to Füser and Georgi (2013), some of the problems of implementing data usage initiatives are related to an absence of relationship between business and IT, the implications that it has in matters of privacy, security, standards definition and with a weak strategy in what concerns the changes needed in order to redirect the enterprises to use data wisely.

In order to reach the so desired insights first, there are some phases to go through (Ernst & Young LLP, 2011):

- Identify and capture potential data opportunities: basically, in this step, it is expected to recognize which data sources might bring value to the enterprise and then how can they be captured;
- Generate insight from data: here the goal is understanding the data and patterns on it;
- Use insight to improve relevance: at this point, the insights gathered must be explored in order to comprehend where they can be used to increase the enterprise's importance and actually add value to it;
- Realize business benefits: as in any other project, it is required that the company presents results so the investment could be justified. So, they must make sure they can measure evolution through this process.

But might this concept be applied when the data becomes big data? Well, this brings up Big Data Intelligence, which is nothing more than transforming BD into insights (like happens for smaller data volumes with DI) and increasing the communication between the different parts involved in this process in order to magnify the base for the data (Dubey, 2018).

6.2. BUSINESS INTELLIGENCE

Sometimes, Data Intelligence is confused with the well-known Business Intelligence (BI). Due to understanding why this happens, it is required to understand what both DI and BI are. Since DI was already introduced is time for Business Intelligence.

Business intelligence can be defined as the use of computer power to help in the decision-making process inside companies, in other words, it is about to use past or present data to future choices (Scheps, 2008).

For Scheps (2008) the best definition for BI is: "Business intelligence is essentially timely, accurate, high-value, and actionable business insights and the work processes and technologies used to obtain them".

According to him if the BI outcome insight misses one of those four characteristics, the solution is a failure. Since they are so important for the case is pertinent to explore a little bit their meaning:

- Accurate Answers: the information presented as a result of a BI process must be accurate so the stakeholders can trust it and relay their decisions over it;
- Valuable Insights: not all information that can be deduced through data has value or impact on the company. That is what this point is made off, in order to BI be successful the insights must have an impact in organizations not just some deduced information;
- On-time Information: the information must be on-time, that means, in that must be available in the time of need not after it. Even good information can be useless if it is not available when needed;
- Actionable Conclusions: practically, this means that it does not matter having conclusions if there is no possible action over it. The only conclusions that matter to organizations are the ones that make them see something and choose a path based on it.

By now, it is possible to understand that the similarities between BI and DI might induce the terminology confusion but also it is possible to tell apart some differences that allow to distinguish them. Looking for the presented definitions becomes easy to affirm that BI consists of understanding the business process and its data; on the other hand, DI uses data for future efforts (Rai, 2016).

In order to deepen this concept, it turns out to be vital to investigate its architecture. According to Scheps (2008), a good architecture document must have information on the following areas (Figure 18):

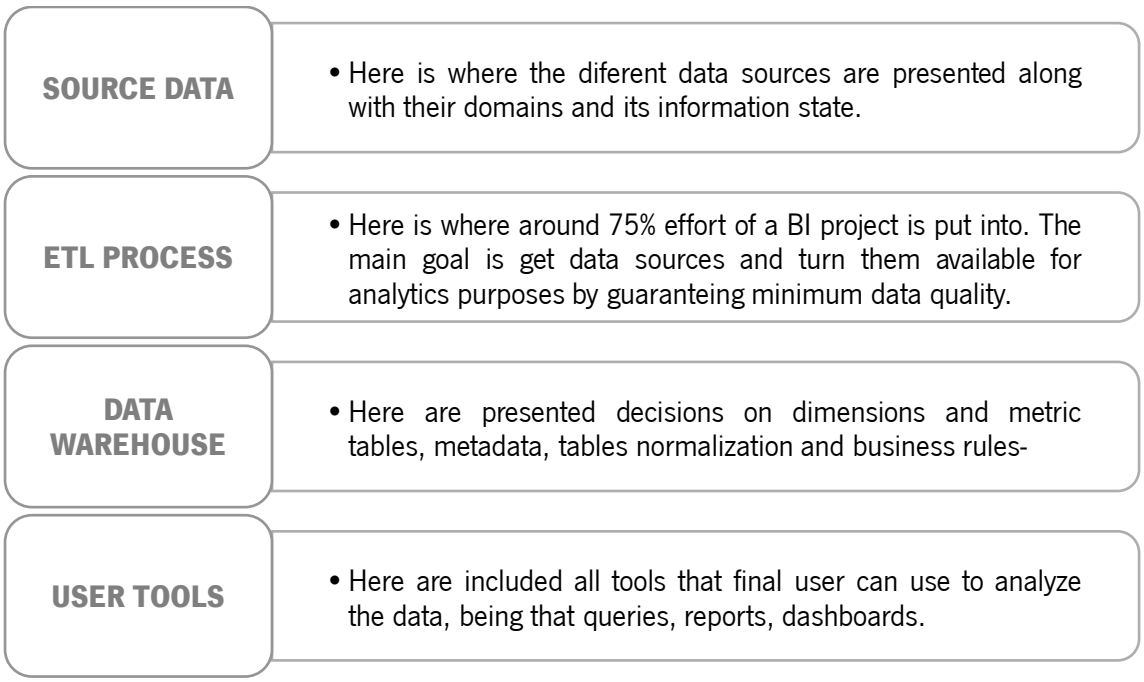


Figure 18 - Business Intelligence Architecture.

In another way, in the beginning, are faced several data sources, being them, usually, operational DB and other external sources. So, it is easy to understand that the fact that data are coming from different sources imply different categorization or structures which need to be solved first in order to allow rich future analysis. Here is where ETL takes place.

ETL is the process of extracting, transforming and loading data from original data sources into the DW or Data Marts. The problem is not moving data but transforming it, which can be referred as "effecting any change on the individual data elements required to create a single, useful, correct body of data to be used by the BI front-end applications" (Scheps, 2008).

Since the main goal is using this data to support decisions, it is vital that in the DW or DM. In order to reach that, it is required to defeat inconsistencies, missing values, integrity constraints and so on. For that it is plausible to call upon three different kinds of tools: Data Migration, which consists into simple transformation rules, Data Scrubbing, used for certain domains and approaches matching techniques, and Data Auditing, which allow to find out rules and pattern on data (those can be considered a variant from DM tools) (Chaudhuri & Dayal, 1997).

Concluded the needed transformations is time to load data into the DW. This does not mean necessarily the end of processing, some things like sorting, aggregation functions, and so on might still happen. These DWs need to be able with much more data than the operational

databases (Chaudhuri & Dayal, 1997). But that does not imply that all data must be in one big DW, it can be spread over several Data Marts connected between themselves (Scheps, 2008).

A data warehouse can be defined as a system that contains all past transactional events to which corporation users have reading access through reports and dashboards (Scheps, 2008).

After all, this work is important to keep in mind that the system needs to be refreshed periodically according to the organization's demands. Some of them might need it in a short time others could do it weekly, for example, others might implement different periods for different groups of data (Chaudhuri & Dayal, 1997).

Filled the Data Warehouse is time analytics. Here the main concept is OLAP which has as the main goal providing to users a set of possibilities to rearrange data to have a multidimensional view over the base data (Scheps, 2008). This is more explored in the next subchapter, OLAP.

At the end of this process, it is possible to establish some analysis, queries, reports or induce data mining processes. This can be clarified by the following schema (Figure 19):

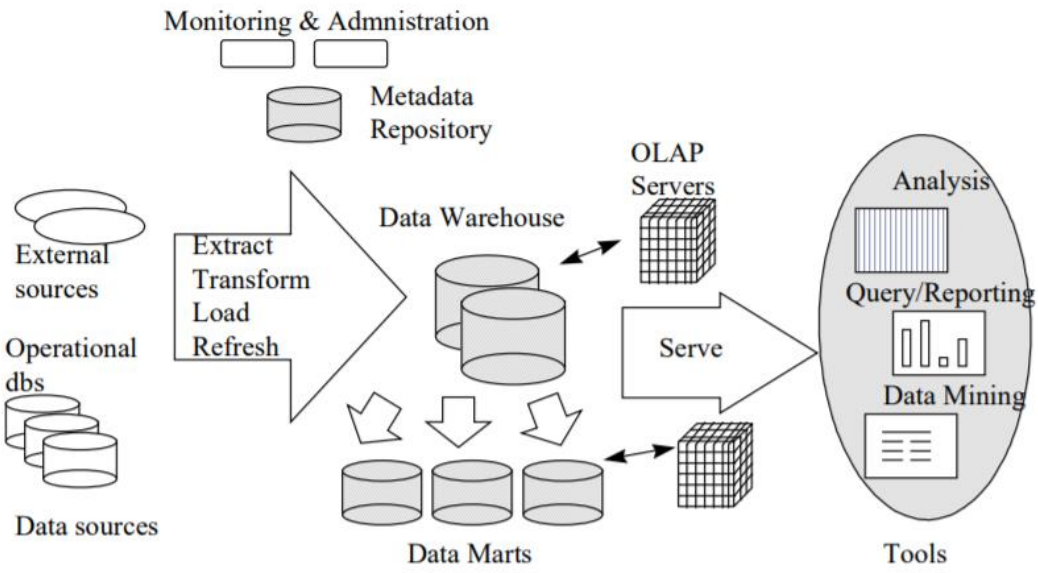


Figure 19 - Data Warehousing Architecture (Chaudhuri & Dayal, 1997).

Furthermore, some principles are followed while architecting a BI system. According to Nedelcu (2013), those are:

- Scalability and High Performance: the magnitude of growth in data, users and applications terms must be “predicted” beyond the actual art state;

- Economies of Scale: the implementations must be conducted by thrift in Information Technologies (IT) operators, servers, and so on. Just guarantee the operability of the system without overspend resources;
- Complete Functionality: the system must include all business intelligence needs for companies without the need for extra integrator effort;
- Incremental Growth: the system must allow its growth in every sense of the word and not imply a new system according to the expansion of the requirements;
- Openness and Extensibility: apart from the decisions taken, functionalities must be accessible through web services APIs (Application Programming Interfaces);
- Centralized Consistency with Distributed Governance and Self-Service: just one consistent output must be shared through the enterprise but also allow the construction of local level consistent solutions;
- Rapid Development and Deployment: design decisions must insist on quick development and deployment of reports and applications;
- Consistent Experience: developers keep trying to give users a full experience by allowing the same BI solutions being used through different user interfaces.

6.3. OLAP

Since On-Line Analytical Processing is positioned in the centre of analytics when the matter is data warehousing gained here particular importance. To start this subchapter is fundamental to dive into what OLAP is really made about and which types exist.

According to Scheps (2008), OLAP is responsible for "allow users to navigate, retrieve and present business data" without putting much effort into writing complex database queries to do so.

For that, OLAP takes the opportunity of being rooted over a multidimensional data model. Data are moulded in cubes of structured uniform facts (measures that are most likely to be numbers) outlined from dimensions where descriptive values stand. Those dimensions create an information bloc where dimensions work as coordinates to find measure values related to them inside the cube (Mansmann & Scholl, 2007). Dimensions serve to give context to those measures and is described by a set of attributes, which might be related between them through a hierarchy (Chaudhuri & Dayal, 1997). Hierarchies allow to introduce new aggregation levels which can bring rich analytical results, but while defining them it is required particular caution because sometimes

there are attributes that cannot be summed up by other, what can lead to parallel hierarchies (ruled by different criteria) (Mansmann & Scholl, 2007). Another main feature of OLAP technology is its capability of aggregate data by one, or more, dimensions (Chaudhuri & Dayal, 1997).

6.3.1. ARCHITECTURES

Naturally, there is more than one architecture proposed when the matter is OLAP. Here is presented the three most known (Scheps, 2008):

- MOLAP or Multidimensional On-Line Analytical Processing: takes in its base cubes and is built for speed. Data are stored in structures whose goal is to accelerate retrieval. For that, a new layer is created between databases and OLAP access tools. This corresponds to the architecture that has been described so far and the most traditional one;
- ROLAP or Relational On-Line Analytical Processing: once RDBMSs had problems relating themselves with OLAP is was required to approach the problem in another way, so, a cubeless OLAP was born to face it. Here appears a semantic layer between databases and OLAP access tools that are supposed to act like a cube in MOLAP. Even though companies can use their already existent databases when complexity starts being increased the ROLAP performance suffers;
- HOLAP or Hybrid On-line Analytical Processing: this one tries to mix both MOLAP and ROLAP. To do so, cubes take place offering rapidness in performance and refreshing while ROLAP components take with it the space-saving characteristic bypassing to OLAP cubes just summed information.

6.3.2. RULES

Another relevant topic to mention is related to Codd's twelve rules for OLAP. So, here are presented those rules in order to provide a wider approach to this theme (Codd, Codd, & Salley, 1993):

- Multidimensional Conceptual View: manipulation and analysis over data must be instinctive, and a business-related multidimensional model must be able on the system;

- Transparency: a front-end system must be associated with the OLAP system;
- Availability: only data required to analysis must be displayed;
- Consistent effort: the effort of the OLAP system must not be according to the number dimensions;
- Client-Server Architecture: OLAP system must be Client-Server type;
- Generic Dimensionality: dimensions must be equivalent in matters of structure and operations;
- Dynamic Treatment of Sparse Matrices: adaptability between physical scheme and analytical model optimizing treatment of sparse matrices is required
- Multi-User Support: the system must allow teams of users and parallel processing as well;
- Unlimited Crosswise Dimensional Operation: a clear distinction between dimensional hierarchy and automatically execute associated calculations;
- Intuitive Manipulation with Data: the system must have an intuitive interface;
- Flexible Declaration: discrepancies in rows and columns disposals must be allowed;
- Unlimited Dimension Number and Aggregate: there should not have any restrictions on what concerns dimensions and aggregations.

6.3.3. FUNCTIONALITIES

But what does OLAP allow to do over data that makes it so interesting? Well, it provides a set of functionalities to exploit the information and see it through different perspectives. In order to organize this subsection, it is presented Table 1, according to Svetlana Mansmann (2007), with several functionalities and a brief description of all of them, for example, the Drill-Down functionality consists in increasing the granularity level along a dimension:

Table 1 - OLAP Functionalities.

FUNCTIONALITY	DESCRIPTION
Drill-Down	It increases the granularity level along a dimension.
Roll-Up	It decreases the granularity level along a dimension.
Drill-Through	It presents the original facts without any aggregation.

FUNCTIONALITY	DESCRIPTION
Drill-Within	Drills down using a different hierarchy in the same dimension.
Drill Anywhere	Drills down a dimension still not used.
Drill-Across	Joins multiple related data cubes by their shared dimensions to combine/compare measures.
Slice & Dice	Selects a sub-cube by applying conditions on several dimensions.
Slice	Filter one of the dimensions to a single value.
Dice	Specifies the values that are excluded from a dimension.
Select	It turns a dimension into a set of values or a range of them.
Filter	Specifies conditions of a selection on a dimension in order to get different aggregations.
Conditional Highlighting	It marks the aggregations according to conditions' satisfaction.
Pivot	Changes dimensional orientation of the view.
Push	Stipulate a measure from a dimension category.
Pull	Convert a measure into a dimension.

All of this is very interesting, but what if it needs to deal not just with structured data but also semi-structured and unstructured data? Can the same architectures be applied? Do the same functionalities make sense in those different types of data? Well, logically no. Somethings might be just adapted others might be completely different.

6.4. REAL-TIME ANALYSIS

This is a topic that has been gained more visibility each day, but why? Well, more and more, organizations need to deal with streaming data and with time-sensitive questions that require real-time analysis.

This real-time data can come from sensors, social media, mobile networks, and so on. This brings a problem along the way. When is needed to act over the actual state of the data, this might imply working in real-time with data that is in movement (Hurwitz, Nugent, Halper, & Kaufman, 2013).

With this in mind, it became important so settle up what is data streaming, complex event processing, and which real-time platforms big data has to offer.

6.4.1. STREAMING DATA

Starting with streaming data, it can be defined as "an analytic computer platform that is focused on speed" (Hurwitz, Nugent, Halper, & Kaufman, 2013), that is related with the need of a continuous stream of data to be handled (analyzed and transformed according to the needs). In other words, and keeping in mind that the value of analysis is corroded by time, streaming data are important in time-sensitive businesses that need data analysis in real-time while data still in motion.

To work like this, the physical environment must be a clustered hardware and with the capacity to work parallelly over the data analysis, which are critical points of this technology. Another critical point of streaming data is the fact that analysis can just be applied once over the data after data being streamed there is nothing else to be done (Hurwitz, Nugent, Halper, & Kaufman, 2013).

6.4.2. COMPLEX EVENT PROCESSING

Identically, Complex Event Processing (CEP) is supposed to deal with data that still in motion but, differently from streaming data it does not analyze huge data volumes in real-time, it "is a technique for tracking, analyzing, and processing data as an event happens" (Hurwitz, Nugent, Halper, & Kaufman, 2013). Basically, it tries to find a connection between streaming and matching patterns with defined behaviours. Its idea passes by events that imply gathering and merging data from different data sources to find out patterns over which users might act.

To the second step, CEP uses EQL, Event Query Language, to execute queries over the data in order to achieve the patterns referred. These systems support query operators' categories such as filters and transformations, window and aggregation operations, join two event streams on an attribute and temporal sequence of event patterns (Simmhan & Perera, 2016).

Thus, the main difference between these two technologies is related to the fact that streaming data are dedicated to huge data volumes analysis in real-time while CEP is dedicated to particular use cases based on events and actions. So, the first one allows instant insights while the

second one allows to stream data and enhance business power by applying business rules to the analysis results (Hurwitz, Nugent, Halper, & Kaufman, 2013).

6.4.3. DISTRIBUTED STREAM PROCESSING

Another technology that has been increasing its popularity in the past few years is DSP, Distributed Stream Processing, once allows real-time analysis of huge data volumes and these tasks can be split over different machines inside a LAN or in Cloud. Furthermore, it offers lower latency than CEP and for that are used to pre-process data, that most of the times include ETL, and pass it to analytical platforms, which can be a CEP engine (Simmhan & Perera, 2016).

This brings some advantages related to reducing costs by reducing data transferred volumes, reducing the latency of sending data to the Cloud a receiving the analytical answer, and increasing the privacy of data sources once they are not sent to the public Cloud because it handles local analysis. For that, DSP has become a field of study spotlighted and increasingly used in the Internet of Things context (Simmhan & Perera, 2016).

6.5. TOOLS

There are tons of tools and approaches related to data analysis. In order to enrich this subchapter, some of those tools are presented and compared. In view of embrace here different technologies, in the way they are used and in their complexity, there were chosen five of the most used and know tools for analytics purposes: Tableau Public, R programming language, Python, RapidMiner and KNIME.

6.5.1. TABLEAU PUBLIC

Tableau Public is a software that allows the connection of any data source, being that a Data Warehouse, Microsoft Excel or web-based data, and then allows the creation of data visualizations, dashboards and so on (Proschool, 2018). This tool is clearly directed for data visualization, without any programming language knowledge required, which makes it much more intuitive and easier to use when compared to other tools in the market (Bose, 2016).

Some of the advantages of this technology are related to the interactive data visualization that can be integrated with blogs and web pages and the fact, already mentioned, that no

programming skills are needed to interact with this tool. This makes Tableau be the best option available in matters of data visualization.

However, some limitations such as little scope for restricted access, the limited OData[18] sources reading ways, but the main one is related to the data size limitation of one million rows.

6.5.2. R PROGRAMMING LANGUAGE

R is a leading analytics tool and often used to statistics and data modelling. This technology allows easy operation of data and a set of different ways to represent it (Proschool, 2018). Like all other technologies, R, as some advantages and limitations important to consider.

In its favour, R has the fact that it can handle objects with no complexity or size limitation. Other than that, it is fully programmable, which allows users to create their own functions and adapt the existing ones according to their needs. Another advantage is related to the fact that the user can process analysis and will always have a visualization of it, regardless of how sophisticated the computation is. Along all of this, a variety of statistics, real-time analysis, clustering, etc. are provided and are highly extensible (Rossiter, 2012).

Against it, are the simpler GUI interaction, since users must type commands to execute analysis and plot the results, step-by-step executions, and the fact that users must learn the critical way of thinking along with new data types so they can develop new methods. Another critical disadvantage is represented by the fact that R tends to consume all the available memory very quickly (Rossiter, 2012).

6.5.3. PYTHON

"Python is an object-oriented scripting language which is easy to read, write and maintain" (Proschool, 2018). This language has been gaining importance once it can be used in an integrated way with SQL Server, MongoDB, and JSON, for example, as can easily deal with text data, and for its libraries.

So, in what concerns Python, it has as main advantages associated with its extensive libraries, since it owns code for several purposes, the fact that can be extended to other languages like C++ and C as it can be embedded in those languages, the simplicity of the coding associated

[18] OData, or Open Data Protocol, is an OASIS standard that includes the best practices for developing and consuming RESTful APIs (OData, 2017).

with its libraries that allows less writing to have more done and it to be learned more easily. Additionally, it is a language quite easy to read and debug, support object-oriented paradigms, and implements WORA[19] as far as no system dependencies features are applied (DATAFLAIR, 2018).

Even though this bright side, there are a few limitations associated with it. First, if the project has an essential requirement speed, this might not be a good choice once Python is read line by line. Second, despite being a good server-side language is rarely used for mobile computing. Another limitation is related to the fact that it a dynamically-typed language, which facilitates the programmer's life but can induce run-time errors. Finally, Python layers to connect with databases still underdeveloped (DATAFLAIR, 2018).

6.5.4. RAPIDMINER

RapidMiner is a powerful and complete data science tool that mixes machine learning, data mining, data visualization, processing, statistical modelling, text analytics, and predictive analysis. This platform has already a good acceptance from the market, and it keeps growing (Proschool, 2018).

To defend its position must be referred to its tolerance to a large set of data source types and its usability to application development, rapid prototyping, training, education, and research. But as the main advantage, it must be referred to its integrated environment for business analytics, predictive analysis, text mining, data mining and machine learning (as referred).

Although, its data size constraints, in a matter of a number of rows, and its high requirements for hardware resources, sometimes make it be put aside along the poor data visualization that provides (Bose, 2016).

6.5.5. KNIME

KNIME is an open-source tool that allows data manipulation, analyze and model trough visual programming. Furthermore, it is often used to integrate data mining and machine learning mechanisms, along with its modular data pipelining concept (Proschool, 2018).

[19] WORA, or Write Once Run Anywhere, allows portability to be implemented without having to code in each platform that it is wanted the code to be executed.

Even though KNIME supports programming languages, it does not need them to work, providing to its user an interface where they just need to drop and drag connections between activities. More than that, this tool can be stretched to run chemistry data, text mining, python and R. In contrast, as a really poor data visualization segment, limiting this technology (Bose, 2016).

6.5.6. CROSS-OVER SUMMARY

In order to turn this comparison more visual, here, it is presented a table (Table 2), to sum up, the information passed previously and easily compare the techniques explored above in this section.

To understand the data "values" it is important to recognized the evaluation scale used in some variables, that was from 1 to 5 (being 1 the minimum and 5 the maximum), So, if a tool is evaluated with a 5 in the "Data Source" matter, that means that as no restrictions, or almost none, concerning the data sources that can be linked to it. On the other hand, if in the matter "Data Volume" it is evaluated with a 3, which means that it has a restriction regarding the volume of data that can be inputted into it. Concerning "Visualization Capabilities", once all the technologies presented allow it, if a tool is classified with a 5 this means that it provides a complete set of data visualization tools, if not, the value attributed denounces some restrictions. Concerning if they have or not Data Mining, Machine Learning and IoT Capabilities, sometimes "-" appears once is not applicable to technology, the other evaluations are higher according to how well they fit these topics. Regarding the topic Programming Languages, the evaluation is higher according to the balance between the versatility of coding and the simplicity of user-friendly interfaces.

The factors approached in the following table have different weights associated with them (according to this project ideals), those variate between 1 and 3 (being 1 the minimum relevance and 3 the maximum) and are indicated in front of each factor between parenthesis.

In other to reach a final "evaluation" for each tool, the value in each parameter is multiplied for the weigh for that parameter and then summed up with the other parameters. This way, it is possible to rank the tools presented in this subsection.

Table 2 - Data Analytics Tools Comparison.

	TABLEAU*	R	PYTHON	RAPIDMINER	KNIME
Data Source (3)	3	5	5	5	5
Data Volume (2)	3	5	5	3	5
Programming Languages (1)	4	3	3	4	5**
Visualization Capabilities (2)	5	3	3	2	1
Data Mining Capabilities (3)	-	5	5	5	5
Machine Learning Capabilities (3)	-	5	5	5	5
IoT Capabilities (2)	-	5	5	4	5
Evaluation	24	74	74	67	72

* Regardless the fact that Tableau is a tool focused in the visualization component, it is here included once it can be combined with other tools or with its Data Mining add-on.

** Even though KNIME does not need any programming language to work, it provides a connection to some languages if the user wants to use them.

After this comparative evaluation between these five technologies, it is possible to see that R and Python offer a quite similar experience in quantitative matters. So, the choice between these two languages would now be held by the fact that python is a more versatile language already in use in IOTech.

7. RELATED WORK

Prior to trying to solve these questions, nothing more logic than search about what already was proposed by others and how it approaches a solution, for the whole problem or parts of it.

First, before jumping into some works, is important to mention KDD, or by other means, knowledge discovery in databases, which is a study area that is commissioned with the development of method and techniques to extract valuable information from raw data no matter what its type is. This can be done by betaking data mining techniques, for example. Applying this process to semi-structured and unstructured data is “a computational challenge for obtaining statistical and different kinds of prediction from a wide range of fields” (Rusu, et al., 2013).

7.1. STRUCTURE IN UNSTRUCTURED DATA

Hereupon, first, it is presented a few works that try to reach some structure in unstructured or semi-structured data, or find some rules to help in the knowledge discovery process.

Rusu (2013), propose trying to add structure to unstructured data so the KDD process could deal with that. To reach such a structure, he proposes some steps to do so. First, the unstructured data must be analyzed. Then it is time for data extraction, which means retrieving information from one medium to another (a wrapper gets the current data and converts it into an XML structured format or data relations, for example). After this, the syntactic analysis takes place, where a parse tree (verb, object) is created so does the structure. At that point, a data classification must happen to categorize data according to some modules. In the end, some rules are inferred in order to reach conclusions on the classified data.

Sint and his colleagues (2009), goes along with the idea of finding a structure in unstructured data. For that, they used techniques of information extraction and natural language processing and made use of tags to describe the text.

Another similar approach to this problem was taken by Lin, Jun, Hongyan, Zhongwei and Zhanfang (2018). In their research, they developed a method so they could extract rules from semi-structured data in order to work over them to extract knowledge. This method of them involved three phases, being the first one “Data Acquisition and Processing”, where semi-structured data was extracted from data sources and converted into structured data, in XML form or relational DB, then appears the “Data Calculation” phase where the minimum support of data, along with its implication strength, are calculated, and to finish, “Rule Extraction” per say, where rules are extracted according to implication relations, and the results are presented.

7.2. STRUCTURED QUERIES OVER UNSTRUCTURED DATA

Another approach involved querying unstructured data by applying structured queries. This can be seen in the next two works presented.

One proposal comes from Liu, Dong, and Havely (2006), that try to apply structured queries to unstructured data. To do so, they extract keywords from structured queries and then search them into data using information retrieval techniques. So, the keywords can be overlapped into unstructured data to acquire more relevant data to the structured query.

Chu, Baid, Chen, Doan, and Naughton (2007), followed the ideals of querying structure into unstructured data. Hence, they proposed a system where users can introduce documents in their raw state and query them having on base some keywords. In their thoughts, these documents would be inserted into a table in an RDBMS that has only two attributes, id, and text and would always be available for querying.

7.3. INTEGRATION BETWEEN OLAP AND INFORMATION RETRIEVAL

A different way to solve this was proposed by Priebe and Pernul (2003), that redirected their efforts trying to find out an ontology-based integration between OLAP and Information Retrieval. Their approach takes on its base a Resource Description Framework (RDF) based ontology that stores metadata from all data sources (in this case, an OLAP system, a Document Management System and a navigate portal content). This allows fuzzy queries over metadata.

7.4. MATURE OLAP OVER SEMI-STRUCTURED DATA

A completely different approach was based on OLAP's well-known structures. The next two works can be included in this perspective.

One is presented by Mansmann, Rehman, Weiler, and Scholl (2012). They based their work on mature OLAP and trying to find which parts of the dataset are plausible to be converted into dimensions or facts, to be ultimately extended using the content-driven discovery of additional characteristics.

Ratav and his colleagues (2007), parlayed on OLAP as well and showed how it could be used to analyze semi-structured data. They proposed a factless multi-dimension and pointed a few relevant operations and aggregation functions.

Based on an already existent system, Wolff (2014), introduced the SP system potential in big data analytics. In his words, this system “has potential as a universal framework for the representation and processing of diverse kinds of knowledge”. His approach consists of discovering “natural” structures in big data, which help in pattern recognition and natural language processing and support realizing analyzes over streaming data.

7.5. OLAP APPLIED TO NoSQL DATA WAREHOUSES

Some other authors invested their time trying to find out how could they apply OLAP cubes over NoSQL Data Warehouses. One of them was Khaled Dehdouh (2016), that proposed a Multi-Cluster Cube that consists of an aggregate operator that builds OLAP cubes for big data DW that use NoSQL columnar DBs. In short, it is used MapReduce to parallelize handling data and aggregate data according to different granularity levels. In the end, to join everything, the MC-CUBE executes two jobs among the other ones necessary to create it.

Another one was handled by Chouder, Rizzi, and Chalal (2017), but this time they approach OLAP over NoSQL document stores. So, they proposed EXODuS, which allows OLAP querying over document stores, it allows users to form multi-dimensional queries linked by OLAP operators and from there get data analysis environment and access to real-time reports.

7.6. REAL-TIME QUERYING

Now, more redirected to real-time querying it is presented as an approach by Liu and his colleagues (2013). They proposed imMens, which is a browser-based system that takes on its base methods that enable real-time interaction by integrating multivariate data tiles (dedicated to pre-processing and loading data in a dynamic way) and parallel processing (allowing interactive visualization of big data).

8. CONCLUSIONS

After exploring the current solutions and ways proposed to solve this problematic, it became clear that there is no solution to solve the problem presented here as a whole. It is important to bear in mind that the purpose of this dissertation is the development of a solution that can deal with and process the data variety induced by big data, in other words, a solution that must be able to sort out all kinds of data (structured, unstructured and semi-structured). After processing this data, the solution must be able to display relevant information (information with value to the users), retrieved from the data, in real-time. To do so, a set of dashboards is used and it must be included in a web platform that allows the users to consult the retrieved information in an

interactive, clear and easy way. This solution should be modular, optimized and adaptable to different realities.

There are several works that try to solve the problem associated with information retrieval using unstructured or semi-structured data, but almost every single one of them fails in dealing with the analysis of this kind of data in useful time. On the other hand, as presented, there are solutions that enable real-time analysis but that is still not the case when it comes to dealing with data other than structured data. Nonetheless, the solution proposed by Chouder and his colleagues (2017), is quite close to what is proposed in this dissertation, once it allows real-time analysis of data originating from document storage. So, what is the difference between the proposal in this dissertation and EXODuS? Well, EXODuS does not allow the final user to interact with the platform where the information is displayed, once it generates reports and not a platform with an interactive set of dashboards. Another difference is the fact that this proposal was developed as a whole prototype meant to be an individual solution, and consequently it is not endued with modular and adaptable characteristics.

Another paper studied is from (González & Berbel, 2014), which suggests a study carried out regarding what already exists to deal with unstructured data while building OLAP systems. For that, they defined five Classes considered relevant in this research (A, Multidimensional Modeling and Data Warehouse design; B, Data extraction, cleansing, transformation and loading; C, Data Warehouse architecture; D, Analytical front-end tools; and E, Maintenance and evolution of Data Warehouses) in order to present which papers (from 42 selected, some of them are also presented here) indicate work in those classes, and are therefore more complete and reliable. Nevertheless, there was not a complete solution (one with work on the five classes defined), based on their research, for the problematic being addressed.

It is important to notice that the work of González and Berbel is from 2014. Consequently, it is important to take into account more recent approaches presented above. So, there two latter solutions were presented, one from Dehdouh in 2016 and one from EXODuS in 2017. Analyzing them through the five classes it is possible to conclude that none of them is complete. In what concerns the first one, there is no work revealed in classes B and D (ETL and Analytical front-end tools) which excludes this solution as a complete one, and EXODuS, as mentioned before, has no workaround front-end tools (missing D class), which rules it out as well.

In this perspective, it is clear that this dissertation's theme has already been studied, as a whole and in parts, and there is already at least one solution to the problematics involved. However,

there is always room for different approaches with other features and capabilities to handle the situation better than done previously. That is why this dissertation has such potential in this theme, for it proposes a solution to the problematic of data variety and useful time analysis regardless of its environment by implementing modular characteristics and adaptability to different realities.

In the next chapters a set of requirements towards the elaboration of the solution proposed is presented, followed by the architecture, and the developments underlying it, to suit all those requirements, aiming to fulfill the insufficiency in this area that has been introduced here.

CHAPTER 3 – METHODOLOGICAL APPROACH

In this third chapter are resented the methodologies used to get this project through. First, it is presented the methodology used in the research component, Design Science Research Methodology, and then the one used in daily through the solution development, SCRUM, specifying its concrete impact on the project.

1. INTRODUCTION

Once this is a scientific project, nothing more logic than rule its developments according to predefined structures that help to reach high-quality results. Therefore, were chosen two different methodologies:

- Design Science Research (DSR): a methodology designed to help to develop scientific projects and its documentation, conducting the all process since the problem identification till the point a solution prototype is acquired;
- SCRUM: an agile methodology that implies defining which artefacts must be developed and in which sprint it must be done, control from time to time (between 1 and 3 weeks usually) what was developed, what is the developments state, and so on.

2. DESIGN SCIENCE RESEARCH METHODOLOGY

When the methodology Design Science Research (DSR) is brought upon the table, it is necessary to mark the fact that the research must shut with the creation of a relevant artefact for an unsolved situation and it must be relevant for the business problem (Peffers, Tuunanen, Rothenberger, & S. Chatterjee, 2007).

Other than this, it is significant to recognize the six steps taken on this methodology here presented according to (Peffers, Tuunanen, Rothenberger, & S. Chatterjee, 2007):

1. **Identify Problem and Motivate:** in this first step, the goal is to define clearly the problem related to the project and justify the motive beyond the work developed. The fact that the value of the project is justifiable motivates all the parts interested in the solution presented. So, to accomplish this step successfully it is required to understand the actual state of the problem, by other words, know the approaches already proposed to solve the

problem, and recognize the importance of this project in the study field. In this dissertation matter, the results of this phase are clearly reflected in the first subchapter of the first chapter of the present document;

2. **Define the Objectives of a Solution:** after the identification of the problem, it is important to define a set of achievable goals in order to guide the work to result in the expected result. These goals can have different natures, quantitative or qualitative, and must be found through the problem definition. In order to fulfil the requirements of this step the current solutions (if it exists) and its efficacy must be known. In what concerns this dissertation, the results of this step are presented in the second subchapter of the first chapter of the present document;
3. **Design and Development:** finished the first two steps, starts the actual development of the solution previously projected. This step includes the goal of the solution, its architecture, and its creation. In this topic, it is also included the theoretical study necessary to the solution development (literature review). Related to this dissertation, this consists of the development of the final solution, by other means, the modular and optimized prototype to execute analysis over different types of data in useful time. During this phase, will be used SCRUM to control the protorype development;
4. **Demonstration:** this step is about testing the functionality of the solution in some proper context. These trials can be made through experiments, simulations, case studies, or others. This means testing not each component of the solution but test it as a complete product. For this case, a proof of concept is done over the solution using IOTech data from their application ioHub;
5. **Evaluation:** after testing the solution it is time to evaluate it. To do so, it is submitted into analysis in order to find out if it goes along with the predefined goals and if it satisfies the necessary conditions to its validation. In order to improve the results, it is possible to go back to the third step. To evaluate this dissertation is needed to question if the solution actually solves the initial problem, by other words, if it really is a modular, optimized and adaptable solution that allows the users to execute analysis over different types of data in useful time. In the future, this could be incremented by adding users' opinion to this evaluation using, for example, the Technology Acceptance Model (TAM);
6. **Communication:** last but not least, communication is the step where the documentation of the project is taken care of. This includes presenting the work in scientific papers. Here

is where the interested and relevant community gets to know the problem, its importance, the solution, and its utility, innovation, and efficiency. In this particular case, this is handled by the report of the dissertation, respective presentations, scientific articles, and a pattent.

In Figure 20, it is possible to see the process iterations beyond the six steps from DSRM. As it is possible to see in the scheme, this methodology presents itself as flexible allowing its users to retreat to previous steps if needed.

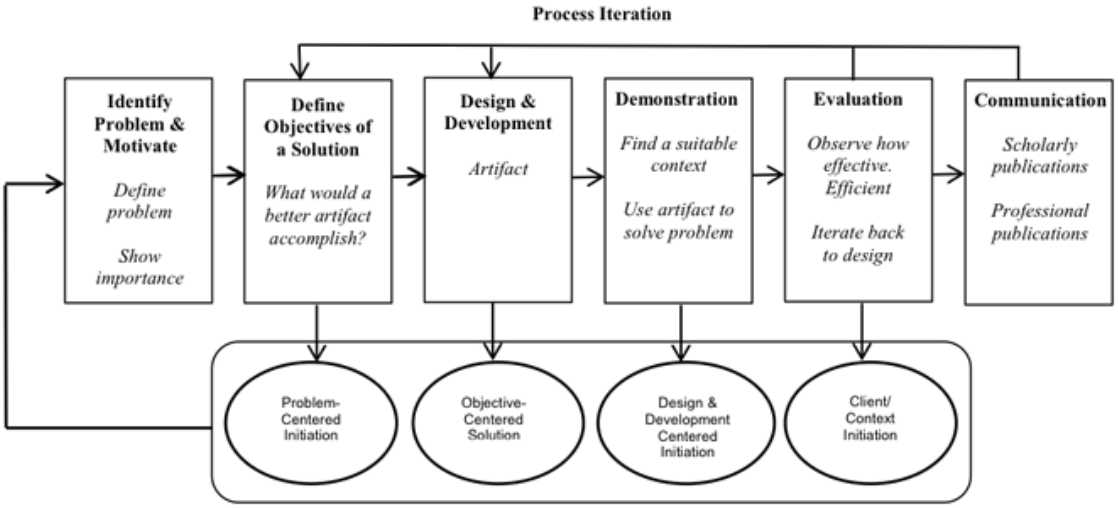


Figure 20 - DSR methodology process model (Adapted from (Ken Peffers, 2007)).

3. SCRUM

This methodology is considered an agile methodology because it is simple and converts the project management into something faster and at the same time-synchronized.

SCRUM's main goal can be defined as “deliver a more suitable product more quickly than with traditional methods” (Cervone, 2011).

In SCRUM, it is possible to find two different cycles, one with a variable duration between one week and one month, called sprint, and another with twenty-four hours of duration, the daily Scrum. It is in the first reunion of each sprint that is defined which requisites will be taken care of in each sprint, “Like projects, Sprints are used to accomplish something. Each Sprint has a goal of what is to be built, a design and flexible plan that will guide building it, the work, and the resultant product increment.” (Schwaber & Sutherland, 2017).

Beyond this first meeting, there are other meetings each week to define which steps will be made in the following week and by who. This helps to reduce variability and uncertainty while planning.

Here, in Figure 21, it is possible to see a representation of the different cycles of SCRUM.

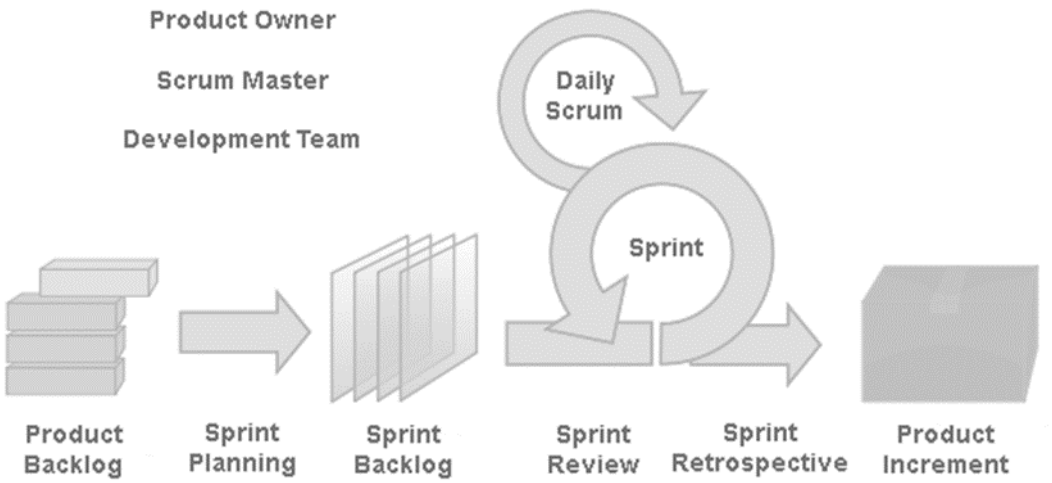


Figure 21 - SCRUM Model.

This methodology is composed of three main components:

- **Roles:** each collaborator as a role associated with him. In the SCRUM context, can be found three roles, the Scrum Master, which is usually associated with the project manager, the Product Owner and the Development team;
- **Processes:** first of all, it is important to mention the kickoff, which is a first meeting where the team discusses and defines the main goals for the project. Then at the beginning of each sprint, another meeting takes place to define the requirements and goals for that specific sprint. Another process of this methodology is the Daily Scrum, which is nothing more than a daily discussion about what was developed during each day. Then appears the sprint review and retrospective. This is the process that includes a meeting to conclude the present sprint. All these meetings are supposed to be something informal with the purpose to increase the work quality each sprint;
- **Artefacts:** here is included the Product Backlog, which is nothing more than a list of product requirements that must be defined according to the project owner's wishes. This list can only be revisited at the end of each sprint, the Sprint Backlog. Another

artefact is the requirements list for each sprint. The Burn Down Charts are the last artefact mentioned here and allows to consult the project evolution during the time.

3.1. PRODUCT BACKLOG

The following table (Table 3) presents the product backlog associated with this dissertation. As mentioned before this artefact must have reflected the project owner's wishes, which in this dissertation case is IOTech. The column Priority serves to sort the requirements according to it, and column Effort represents an estimative on the effort that is needed to accomplish the requirement. Both variables vary between 1 and 5 (being 1 the minimum and 5 the maximum). A third variable was included in order to identify the final state of each requirement. This is distributed between 0 and 2, being 0 the translation for not implemented, 1 for partially achieved and 2 for completely achieved.

Table 3 - Product Backlog.

ID	REQUIREMENTS	PRIORITY	EFFORT	ACHIEVED
1	Data Analysis and Selection.	2	2	2
2	Multidimensional Model.	3	3	2
3	OLAP layer.	5	4	2
4	Main Web Application.	4	2	2
5	Set of relevant Dashboards.	4	3	1
6	Python Restful API: Get Data from the Database	3	3	2
7	Python Restful API: Data Treatment	4	4	2
8	Python Restful API: Retrieve Information	5	5	2
9	Python Restful API: Refresh System	1	2	0
10	Cashing system.	1	5	2

As it is possible to verify in Table 3, some of the requirements were not achieved or at least not on its totality. Regarding the fifth requirement, it is classified with 1 because, even though a set of dashboards were actually developed they have not achieved the desired quality or complexity, and they do not reflect the defined KPIs due to the lack of proper data. Concerning the ninth one,

it was not implemented at all once it has a lower priority associated with it and the time dedicated to this project was needed to improve priority features.

It is important to mention that the tenth requirement was implemented regardless of its lower priority because it adds an innovation component to this project, once there were no records on the attempt of developing such a system like that allows its manipulation while offline. This leads to scientific content creation liable to intellectual protection and a substantial project value increase.

3.2. SPRINT BACKLOG

The following table, Table 4, presents the sprint backlog associated with this dissertation. So, considering the product backlog previously defined, a set of sprints were generated and are here presented, followed by its start and end dates chronologically ordered.

It is important to note that every and each sprint must have a period of time associated that variates between one week and one month, and all of them correspond to the practical component of the dissertation, once this methodology is just used in that part. When overlapped with DSR methodology, all these sprints take place in its third phase (Design and Development).

Table 4 - Sprint Backlog.

ID	SPRINTS	START	END
1	Explore Technologies to be used	05/02/2019	01/03/2019
2	Analyze and Select Data and Development of the Multidimensional Model	04/03/2019	08/03/2019
3	Development of a Python Restful API: Get Data from the Database and Handle Unstructured Data	11/03/2019	29/03/2019
4	Development of a Python Restful API: Data Treatment	01/04/2019	19/04/2019
5	Development of a Python Restful API: Retrieve Information	22/04/2019	03/05/2019
6	Development of an OLAP layer	06/05/2019	17/05/2019
7	Test charts possibilities to include in the Dashboards	20/05/2019	14/06/2019
8	Development of a Cashing System: IndexedDB definition	17/06/2019	05/07/2019
9	Development of a Cashing System: Filters Mechanism	08/07/2019	19/07/2019
10	Development of a Cashing System: Filters Mechanism (cont.)	30/07/2019	09/08/2019

ID	SPRINTS	START	END
11	Development of a set of relevant charts	12/08/2019	23/08/2019
12	Development of a set of relevant charts (cont.) and of the main Web Application	02/09/2019	20/09/2019
13	Development of a set of relevant Dashboards and of the main Web Application	23/09/2019	11/10/2019
14	Revision and Adaption of the Solution	14/10/2019	21/10/2019

4. CONCLUSIONS

Concluded this chapter, it is possible to have a clear vision over how these methodologies help to develop this project, and in which part and phase of it they are present and why.

Moreover, the product backlog defined through the SCRUM methodology helps to understand what the final solution must include, turning it easier to evaluate the project success by the achievement or not the different requirements there described.

Along with it is also possible to check a timeline for those different features to be developed.

CHAPTER 4 – PRACTICAL WORK

In this chapter a vision of the system proposed as a whole is provided in an architecture form in order to make clear how the different outputs of this project are integrated, creating a proper solution to address the problem stated, which is also described here into development layers.

1. INTRODUCTION

This is the dissertation chapter where actual practical developments properly explained can be found. For that to happen, first it is necessary to understand which technologies and tools would help in this process and what they do exactly. Therefore, a complete list of all materials and methods used is presented complemented with proper justification for their usage. Moreover, essential as always, it is also included here an evaluation between different technologies to suit some purpose, justifying the one chosen to handle this project's requirements.

Since the resources used have been presented, it is time to understand the architecture underlying this dissertation and what each layer means and aggregates to the final result.

Well, the architecture provides an explanation on the project's complexity and requirements, but it becomes crucial to understand in practical terms what was really developed to contribute to the realization of each architecture level, and for that four development layers explaining the different mechanisms implemented to carry each component of the solution offered are presented.

A case study, IOHub Companies, which is built based on data provided by the company IOTech, in order to turn testing the solution applied to a case in the society of services into something possible, is also associated with this dissertation.

2. MATERIALS AND METHODS

Here are presented the technologies that were studied in this project context and are used through solution development along with the tools needed to give the proper support to these technologies. In Table 5, it is possible to check all of these technologies and tools followed by a

usage justification. Further, in this report, it is also possible to consult a brief explanation of every single one of them.

Table 5 - Materials and Methods Usage Justification.

		USAGE JUSTIFICATION	TYPE
TECHNOLOGIES	MongoDB	Due to some data sources storage.	Database
	MySQL	To support the multidimensional model.	Database
	IndexedDB	To store the queries needed to fulfil the charts included in the dashboards even without the network available.	Database (API)
	JavaScript	Used as the base programming language to develop all frontend components.	Programming Language
	Python	To develop the RESTful API dedicated to the ETL process.	Programming Language
	NPM	To manage the libraries and packages dependencies through the project.	Package Manager
	Flask	To help to orient the RESTful API in a web way.	Framework
	Cors	To handle cross-origin requests.	Method
	Vue js	To develop the web application.	Framework
	Express	To handle queries' requests to cubejs.	Framework
	Vuetify	To improve the look of the web application according to Material Design.	Framework
	Pandas	To allow pandas dataframe used to store data trough some ETL process steps.	Library
	Pymongo	To establish a connection with Mongo DB database and the ETL API.	Library
	Numpy	To allow numpy arrays usage to store data trough some ETL process steps.	Library
	Google Maps API	To draw and populate maps to be included in the dashboards.	Application Programming Interface (API)

		USAGE JUSTIFICATION	TYPE
	Apexcharts	To draw the majority of charts included in the dashboards	Library
	Cube JS	To develop the OLAP layer.	Library
TOOLS	Robo 3T	Interface to manage the MongoDB database.	Interface
	MySQL Workbench	Interface to manage the MySQL database.	Interface
	WebStorm	IDE used to develop the frontend components.	Integrated Development Environment (IDE)
	PyCharm	IDE used to develop the ETL process.	IDE
	Postman	To test the APIs routing.	API Development Environment (ADE)

2.1. MONGODB

Well, MongoDB[20] was the database chosen to carry this project for the simple reason of being the one used in the enterprise, IOTech.

MongoDB is a remarkable database in what concerns NoSQL databases and document storage, once it combines features along with scalability, having as examples for that range queries, secondary indexes, aggregations, sorting, and geospatial indexes (Chodorow, 2013).

2.2. MYSQL

MySQL[21] was the database chosen to support the data warehouse underlying this project needs, once it is a technology already in use inside the company, IOTech.

Nay, the request of being an open-source relational database (based on SQL) management system would be fulfilled, allowing stabilization of a data structure further into the system reducing uniformization conflicts in the future.

[20] https://www.mongodb.com/
[21] https://www.mysql.com/

2.3. INDEXEDDB

IndexedDB[22] is a recent database API that aims to satisfy requests of data through web applications even when the user is offline. This technology bases itself on key-value data management using transactional databases to store those keys and respective values and the ways to reach it accordingly.

This technology was chosen to support the development of the proposed solution once it can increase the value of it by allowing data queries in a more efficient way (given that is implemented in the application side) along with offline data visualization.

2.4. JAVASCRIPT

JavaScript[23] is one of the most used programming languages in the world wide web context. It is a high-level multiparadigm scripting language that supports event-driven, functional, and imperative programming styles. Therefore, it presented itself as a valuable asset to developing the needed intermediate mechanisms, even more, once a JavaScript framework (Vue.js) is being used to develop the frontend layer of this system, and it is a language already in use inside the company.

2.5. PYTHON

Python[24] is an object-oriented and high-level programming language that has been growing in an exponential way. This language revealed itself as a powerful tool covering various areas, such as web development, machine learning, and data science, for that and for the fact that was the language already in use in the IOTech's context, was chosen to carry on this project.

2.6. NPM

NPM[25], Node Package Manager, appears here to suppress the need for a tool capable of dealing with the enormous number of libraries and packages that are used along with the project,

[22] https://www.w3.org/TR/IndexedDB-2
[23] https://www.javascript.com/
[24] https://www.python.org/
[25] https://www.npmjs.com/

along with all the dependencies that this brings. In order to execute the project, it is required the installation of Node.js which brings along the installation of NPM, facilitating the choice in this matter, adding on its favour the fact that it is an extremely documented technology.

2.7. FLASK

Flask[26] is a Python web framework that is based in other libraries and templates like Werkzeug, Jinja, MarkupSafe and its dangerous. Basically, this microframework aims to help developing web applications saving time in the way and maintaining a simple core making it easier to understand and work with but still possible to be extended.

2.8. CORS

CORS[27], or Cross-Origin Resource Sharing, is the technology based on HTTP headers used throughout this project to handle HTTP requests between different origins (the web application and the requested resources), domains, protocols or ports, by executing a cross-origin HTTP request.

2.9. VUE.JS

The decision of using Vue.js[28] became along with the idea of learning something new and introduce emerging technologies into the proposed solution that might help better its development.

Vue is a progressive framework used in user interface development, being focused on the view layer. This technology was designed so it could be incrementally adopted and easily integrated with other libraries and projects. Furthermore, Vue is known for its power in what concerns single-page applications and assumes knowledge on HTML, CSS, and JavaScript once it is built by joining "pieces" of these three different languages.

[26] http://flask.pocoo.org/
[27] https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS
[28] https://vuejs.org/

2.10. EXPRESS

Express[29] is a web framework that includes a set of features that aim to help developing web and mobile applications along with APIs. In this project context, it became handfull in a queries' management way.

2.11. VUETIFY

Vuetify[30] appeared as a technology required for this project development once the template used in the base of the construction of the application has some components developed based on Vuetify. This is a progressive material design component framework that has as main goal the code cleanliness and reusability to make faster and better Vue applications.

2.12. PANDAS

Pandas[31] is a Python library that provides functions to deal with data structures in an easy way and not disregarding the performance. So, this library appears aiming to fill the breach in what concerns data analysis and modelling.

When this technology comes around the most common concept appearing is DataFrame, which is a pandas' object that allows easy and fast manipulations over data. The data present in these objects can originate from different data files, such as CSV and JSON, and be "converted" to this structure.

2.13. PYMONGO

Pymongo[32] is a Python library that has in its definition a set of functions that aims to handle MongoDB connection and data flow through Python applications that work with MongoDB storage, whether extraction or insertion of it.

[29] https://expressjs.com/
[30] https://vuetifyjs.com/
[31] https://pandas.pydata.org/
[32] https://api.mongodb.com/python/current/

2.14. NUMPY

NumPy[33] is a well-known Python library for scientific computing. More than including an N-dimensional array, known for its power, it also counts with several relevant functions, particularly in the mathematics field, integration possibilities with C, C++ and Fortran code.

Furthermore, this library includes capabilities to be integrated with several different databases and allows the definition of arbitrary data types, increasing the code flexibility.

2.15. GOOGLE MAPS API

Google Maps API[34], is a powerful tool made available by Google that aims the creation and integration of maps in other applications. The reason that made this technology being considered for this project is related to the need for the development of dashboards that must include maps for the analysis required to be better achieved.

2.16. CHART LIBRARY

So as to develop appealing dashboards with relevant, eye-catching and easy to understand charts, it became useful to choose a chart library that would help to do so and keep the different views consistent. To know which one would be better among several options, a study was previously realized, whose results and variables are displayed in Table 6.

First some relevant criteria considering this project's needs were defined, as well as usual criteria in similar projects, and that can be seen in the first column followed by its weight (from 1 to 3, 1 being the minimum and 3 the maximum). Next to it, seven possibilities to do the job were considered, one per each next column, and compared based on a punctuation between 1 and 5 (1 being the minimum and 5 the maximum), with exception for the criteria "Open Source", which is a yes or no answer and an exclusive factor. In the end, the total punctuation is calculated (by multiplying the criteria's punctuation by their weight), and the results are displayed. For example, to obtain the Apexchart result, the calculation was: 5x2+3x3+4x2+5x3+5x1+4x2+5x3 = 70 (the order of values in each parcel of the calculation is in the form evaluation x weight).

[33] http://www.numpy.org/
[34] https://developers.google.com/maps/

Table 6 - Chart Libraries Comparison.

CRITERIA	AMCHARTS	APEXCHART	CHART.JS	D3	GOOGLE CHARTS	NVD3	X-CHARTS
Documentation (2)	5	5	5	5	5	4	3
Drill Down/Roll Up Functions (3)	5	3	1	4	4	3	1
Interactivity (2)	5	4	3	4	3	4	2
Open Source (*)	No	Yes	Yes	Yes	Yes	Yes	Yes
Reactivity (3)	5	5	5	4	5	5	2
Variety (1)	5	5	5	5	4	4	3
Visualization (2)	5	4	4	3	3	4	3
Vue Integration (3)	5	5	5	3	5	5	5
Evaluation	80	70	62	62	68	67	43

*If this request is not full filled the technology is put aside.

So, in Table 6, looking at the comparison of a few charts' libraries and frameworks that were studied, it is by far correct to say that AmCharts would be the best choice for this project. Well, there is a problem in that conception, AmCharts requires a paid license to freely work over it. Following the logic of the most scored technology, pursuing on to the most complete according to the requirements presented, the next in line is ApexCharts, for it is the one being used in what concerns the dashboard charts' development, moreover it satisfies all of the project's requirements and has no negative evaluations. Still, it is important to bear in mind that all components with a lower evaluation are compensated with other technologies, because, for example, Google Maps API helps in matters of the variety of the dashboards presented.

APEXCHARTS

ApexCharts is an open-source JavaScript library that aims at developing interactive charts and dashboards for web pages. Beyond providing theme palettes and styling elements to create pleasant charts, it can be categorized as responsive, once it updates itself according to the screen size in which is the graphics are included. As mentioned, it is referred to as an interactive and dynamic library, that comes with the possibilities of zooming, scrolling, framing into different data series, selecting information based on another chart and so on. Furthermore, ApexCharts also

promotes high performance and some animations to can be displayed while changing between datasets and loading data.

2.17. OLAP LIBRARY

Before starting to build the OLAP layer, it is crucial to decide which technology is going to be used (considering existing ones, their features and the possibility of building a new one according to the project's needs) and on which side of the architecture it takes place: in the Python RESTful API or in JavaScript directly connected to the Vue interface.

After doing some research on what already exists, the possibility of building a new one was ruled out once it would be redundant and a "waste of time" considering that none or almost no new features would be added and probably if so, in a similar way.

As a result of the research just mentioned, a few names popped up, such as Data Brewery OLAP framework and Olapy API, on the Python side, and Cube.js and OLAP cube.js on the JavaScript side. Other, less relevant, technologies were identified, both on the JavaScript side: Square Cube, which got deprecated several years ago even though it can still be used, and cube.ui, that is not a good choice for this project once it is especially directed to mobile development.

2.17.1. DATA BREWERY

Data Brewery is composed of different Python frameworks designed to analyze and process, in other words, and it has components designed to execute ETL processes and more advanced OLAP structures.

Here only a part of the Data Brewery is considered, Cubes, which is a framework and group of tools that aims at developing analytical and reporting applications, OLAP structures along with multidimensional analysis and aggregation searches through data.

This component allows the implementation of both star and snowflake schemas, together with several database models, models whose schema does not come straight from the database and so on.

Concerning the database support, many different databases are allowed to be connected throughout the process, both non-SQL and SQL, such as MongoDB, Microsoft SQL Server, MySQL,

PostgreSQL, Oracle, and a few more. Additionally, it is possible to establish a connection with Google Analytics, Slicer and Mixpanel.

2.17.2. OLAPY

OlaPy is a Python OLAP engine that allows, similarly to Data Brewery, multidimensional analysis, aggregations through data with MDX and XMLA support and ultimately the development of reporting and analytical applications. This technology was built being oriented to sustain aggregations and real-time computation. It also includes an ETL layer, but it is not important in this phase once the data arrives already properly treated and filtered.

Regarding the support given in matters of database connection, it allows connection to some of the most common ones, namely Postgres, MySQL, Oracle, and SQL Server. It also allows data import from CSV files.

Concerning the schemas' possibilities, this technology presents itself a bit more restrictive than the previous one, once it forces the table to follow a star schema.

2.17.3. OLAP CUBE.JS

OLAP cube.js is a JavaScript set of tools oriented to execute data analysis. Therefore, this library helps to extract data and reorder it in order to allow aggregations with the ultimate goal of helping in the decision-making process.

When compared to previous technologies in what concerns databases' connections, this one has a huge gap once it does not allow direct connection to them. Instead, it locally defines an n-dimensional database (written in JavaScript), oriented to support aggregations and chart development.

Regarding the schemas' possibilities, it allows both star and snowflake schemas, with a preference for the second one. Nonetheless, it is a little restrictive, for it forces all dimensions to include at least one hierarchy. That is not a problem for many dimensions, but when dimensions that hold information on coordinates, for example, are considered, this task becomes tricky, once both latitude and longitude (and even elevation, when taken into account) work not hierarchically but as a group.

Like in cube.js, here it is also important to understand how easy and direct a connection with Vue can be. Well, unlike the other one, this technology is pure JavaScript, so the connection is possible but needs to be fully newly developed.

2.17.4. CUBE.JS

Cube.js is a framework designed to build analytical web applications; these can be internal business intelligence tools or analytics tools to be integrated into an existing application. This is a modular solution that includes transformation, data warehouse modelling, querying and cashing API gateway management and user interface development modules.

With regard to databases' connections, cube.js it is quite permissive allowing connection to several of the most used ones, such as MySQL, SQL Server, PostgreSQL, MongoDB, and so on.

Adjacent to it, it is important to consider the comprehensiveness in what concerns the possible schemas. At this point, this technology is very malleable, allowing schemas to be manually defined directly into the code and schema imports directly from the database. Henceforth, this variety of possibilities enables not only simple OLAP functions, but also complex analysis.

Once on the JavaScript side, it is important to understand if this solution has a direct connection with VUE or if it is purely JavaScript. Well, all schemas and connections are established in pure JavaScript, though cube.js provides a Vue wrapper and client-side API.

2.17.5. COMPARISON

So, first the four technologies presented were selected to be submitted into a deeper study, and now they must be compared by crossing them over to some relevant aspects to the project. Similar to previous cases, bellow, in Table 7, a set of relevant aspects to be considered is exemplified, along with a weight (between 1 and 3, 1 being the less relevant and 3 the most relevant), and all technologies presented are evaluated in each of those aspects in a scale from 1 to 3 (1 being the minimum and 3 the maximum). To clarify, the first aspect presented, "Ease to learn/work with", relates to how much available information there is, how clarifying that same information is, how "clean" coding using a specific solution is, and how complex the solution and its connections are compared to what is needed. For that aspect, it is possible to see that, on the Python side, Data Brewery acquired points and Olapy one point, and on the JavaScript side, Cube.js

acquired points, when OLAP Cube.js ensured 3 points. For example, to obtain the Cube.js classification, the calculation was 3x1+3x2+3x3+3x2 = 24 (the order of values in each parcel of the calculation is in the form of evaluation x weigh).

Table 7 - OLAP Libraries Comparison.

TECHNOLOGY	PYTHON		JAVASCRIPT	
	DATA BREWERY	OLAPY	OLAP CUBE.JS	CUBE.JS
Ease to learn/ work with (1)	3	2	2	3
Set of functions available (2)	3	2	2	3
Databases supported (3)	3	2	1	3
Schemas supported (2)	2	1	2	3
Evaluation	22	14	13	24

Given that, Cube.js is presented as complete and adjustable to several realities, as a whole or partially. Furthermore, before the comparison presented, it was also taken into consideration that this layer would benefit in speed matters by being developed on the Vue side, and therefore using Javascript.

2.18. ROBO 3T

Robo 3T[35], or Robomongo, is a lightweight interface for MongoDB usage. So, this tool provides support to Mongo 4.0 and allows functionalities such as data visualization, data querying, import, export, code generation, among others.

2.19. MYSQL WORKBENCH

[35] https://robomongo.org/

MySQL Workbench[36] is an interface for MySQL database usage. This tool is fully equipped with data modelling, SQL development, and several administration tools for data, server and user management.

2.20. WEBSTORM

Nothing fairer than starting this list by explaining the IDE, Integrated Development Environment, used to develop the application. Before taking a decision of which IDE would be used was made a proper research on which IDEs support the language used, JavaScript, more properly the emerging library Vue.js. Beyond this factor, this decision was a matter of preference and counselling. So, after considering Visual Studio and WebStorm[37], the final decision pointed to the second one, taking into consideration its intelligent code completion, error detection, among other features.

2.21. PYCHARM

Equally important is the IDE used to develop the API that holds the data processing through this project. The decision of which IDE should be used to develop this component stood by what the enterprise already uses when it is needed python-support. Therefore, PyCharm[38] was chosen. As Webstorm, Pycharm is a product from JetBrains that has in its features intelligent code assistance, more than python it provides support in what concerns web development, has also included a bunch of scientific tools, such as NumPy (that is presented ahead), and more.

2.22. POSTMAN

Postman[39] is an ADE, API Development Environment, that is flexible in what concerns its integration with the software development cycle. In short, this tool allows test RESTful APIs, regardless of the source, without having to write a bunch of code for that, so that is the role of this tool in this project.

[36] https://www.mysql.com/products/workbench/
[37] https://www.jetbrains.com/webstorm/
[38] https://www.jetbrains.com/pycharm/
[39] https://www.getpostman.com/

3. SOLUTION ARCHITECTURE

As mentioned above, in this chapter, a solution basis architecture is presented in order to provide a full explanation on how the different "phases" of this project add value to the final solution, how they are connected and also their main goals, so as to achieve a proper solution to the problem being studied.

So, in Figure 22, it is possible to see this representation of the solution and the sequence implied through its different phases. Five layers can be clearly identified: the Database(s) that feed the system, the API that processes data in order to retrieve information, the Data Warehouse where the filled multidimensional model is stored, the OLAP layer to query the data and provide different perspectives over it, and, finally, the Visualization, where the data becomes available to the final user through a set of dashboards displayed in an application.

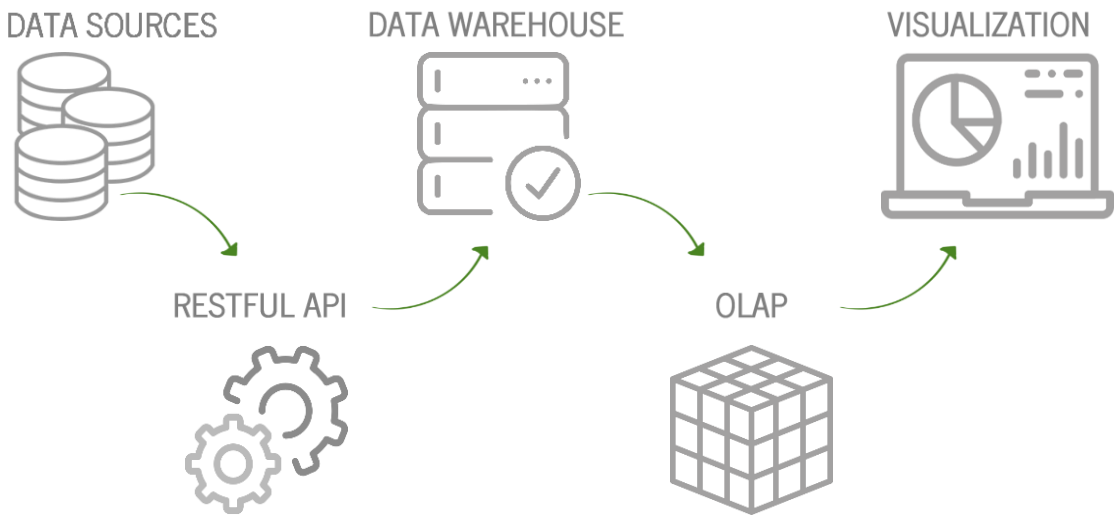


Figure 22 - Solution Architecture.

3.1. DATA SOURCES

This layer represents the database(s) and other sources that provide data to the whole system. In real life, and considering the case study under examination, that is the IOTech database, which is a NoSQL one, particularly the well-known document store MongoDB and a few complementary data sources.

3.2. RESTFUL API

This second layer involves the most important work done over this dissertation. Here is where the data processing takes place, that is, transforming the raw data stored in the previously presented data sources into relevant information to fulfill the OLAP layer.

Basically, it is in this phase that mechanisms are developed to deal with the data variety in order to turn it into useful and workable data. For that, in the first place, it is required to proceed to the extraction of the data from the database, and therefore it must be treated according to the flaws previously detected while making an extensive data analysis, by addressing each situation properly and bearing in mind the data context. This having been done, it is time to work on retrieving information from the data treated and look for the best way to reach that relevant information.

3.3. DATA WAREHOUSE

In the first version of the solution's architecture, the Data Warehouse was not considered as an "independent" layer, its questions underlying the API ones.

After taking a closer look at the architecture proposed previously and the needs that the project's goal implies, it became relevant to address speed and storage matters properly to develop this third layer. Moreover, the question of maintaining the application's dashboards filled with the most updated information even without internet connection led to a recent database hypothesis to support the Data Warehouse on the application, IndexedDB.

So, this layer is about getting the multidimensional model filled in the final step of the API phase and transferring it to the data warehouse database.

3.4. OLAP LAYER

All of the (data) structure-defining process is taken care of in the application, in its underlayer, along with all functionalities. To enlighten this idea, in the architecture presented in Figure 22, it is possible to see that this phase follows the Data Warehouse and precedes the Visualization phase, once it is built over the data stored in the data warehouse and its results are applied only in the Visualization phase.

So, as mentioned, in this phase, first the modulation of the data is included, so then it can be imported into an OLAP structure with proper elements.

Once an OLAP structure is fulfilled, it is time to define its functions, such as drill-down, roll-up, slice and dice, and so on.

3.5. VISUALIZATION

This is the last "phase" of the presented architecture, and as the name suggests, it involves the user's perception of the value of the work beyond this phase. That makes this phase extremely important since if the solution's value is not perceived, the whole solution is a failure, no matter what is beyond it.

So, as mentioned in the previous topic, before jumping into the dashboards, a set of functionalities must be developed in order to provide interactivity and a sense of perspective over data while consulting the dashboards.

That having been achieved, it is time to dedicate to the development of relevant dashboards that reunite the information retrieved and display it in an easy and pleasant way to the users. Both these two steps are included in the application's development.

4. SOLUTION DEVELOPMENT LAYERS

In this second section, the different development layers underneath the previously mentioned architecture are presented. These layers represent nothing more than groups of phases related through high-level architecture, which can be clearly observed in Figure 23. Each development layer is fully explained below, along with some examples (retrieved from the case study presented in the section Case Study: IOHub Companies) considered necessary to understand the system developed.

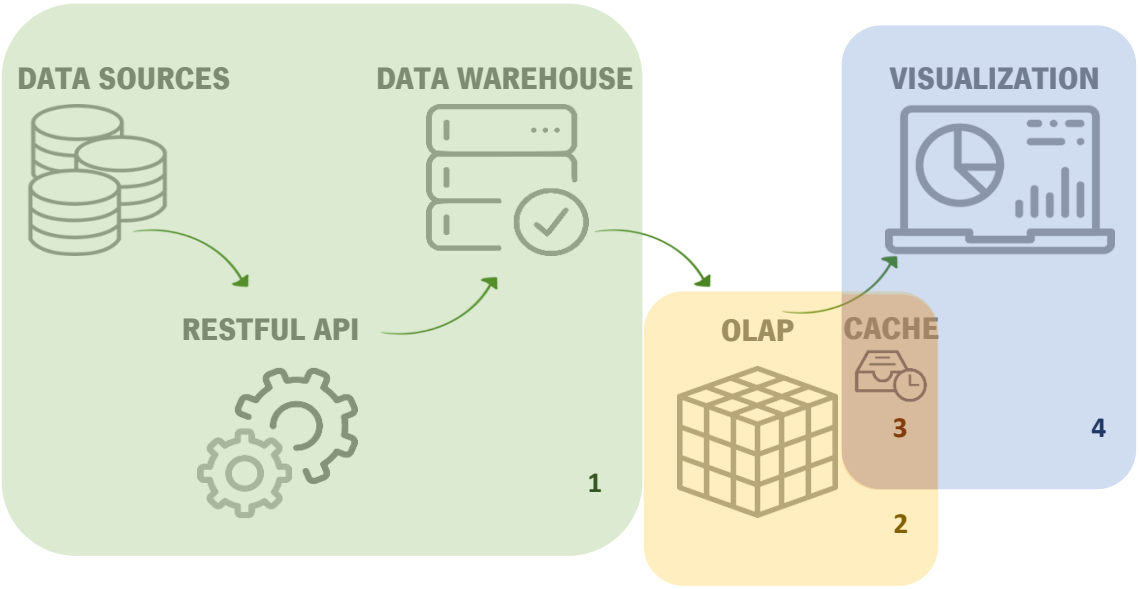


Figure 23 - Solution Development Layers.

As is shown in the previous figure (Figure 23), there are four development layers: Data Development Layer, Analysis Development Layer, Caching Development Layer and Visualization Development Layer.

- **Data Development Layer** (Layer 1)**:** in this first layer all the mechanisms needed to lead data from the data sources until the data warehouse, passing by an ETL process that aims at improving data quality, are included. Here the Data Sources, the RESTful API and Data Warehouse presented previously as part of the main architecture are involved;
- **Analysis Development Layer** (Layer 2): in this layer the mechanisms underlying the OLAP layer construction are included, and therefore this phase of the high-level architecture is represented here;
- **Caching Development Layer** (Layer 3): this layer does not include a high-level architecture component but can be seen as a bridge between the OLAP layer and the Visualization layer. This middle layer includes the mechanisms that make it possible to have a "cache database" that allows offline limited visualizations through data;
- **Visualization Development Layer** (Layer 4): last but not least, the Visualization Development Layer corresponds directly to the Visualization layer presented in the main architecture. Here all of the dashboards' construction and their integration in the web application are included.

4.1. DATA DEVELOPMENT LAYER

This first layer includes all the processes needed to handle the data used as input into the system. In Figure 24, it is possible to verify that this development layer includes the first three phases of the high-level architecture presented previously (Data Sources, RESTful API, and Data Warehouse).

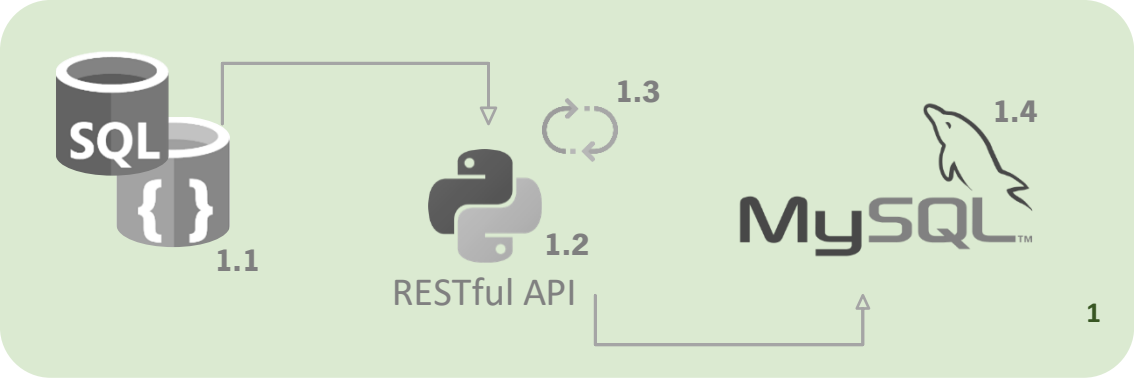


Figure 24 - Data Development Layer.

Regarding the Data Sources phase of the architecture, it is possible to see a direct match between the SQL and NoSQL databases represented here (1.1). During the development of this project only NoSQL inputs, particularly, MongoDB databases and JSON files were used, but the system is prepared to receive SQL databases as well.

As it is completely understandable, these data must be treated before any other operations are done over them. Therefore, a RESTful API was developed (1.2), using Python, to handle all treatments required along with data standardization to properly fulfill the analysis needed. To achieve one system that could be maximized, and well-balanced, in matters of data accuracy and veracity, and the performance of the whole process, several trials were made progressing to a stable and acceptable version of it.

So, in other words, this API must take care of some tasks, such as:

- Data Extraction (from all data sources, the main one composed of the relevant facts' information, and the secondary that helps the model to become consistent);
- Data Cleaning (exclusion of irrelevant attributes and dubious incorrigible values among the relevant ones);

- Data Treatment (correction of the dubious attributes whose information can be crossed with other more accurate sources);
- Data Standardization (in order to link the dimensions' information to the facts);
- Data Load (into the Data Warehouse).

To sum up, five different cases were implemented and tested according to code cleanliness, comprehensive data structure, data fixedness to the following operations in the system, the least possible time since data arrival and its loading, different data (structured and unstructured) inputs' support, and scalability proclivity. The differences between those versions rely on how the data are handled during the process, more specifically in matters of data structure.

The first attempt consists in modelling based uniquely on arrays throughout the whole process, in the second one all arrays were replaced by dataframes. These two approaches are in all matters quite similar once the data are called into some structure inside the API code, handled there, and only when all operations are concluded are the data sent to the data warehouse, "freeing" the structures inside the code.

The results not being satisfactory, another approach was undertaken, changing the placement of the data during the treatment and standardization process. To do so, an intermediate NoSQL database was created, in MongoDB, and as soon as all dimensions (handled with arrays) are ready, they are sent to this staging area from where data are constantly queried in order to fulfill both bridge and facts' tables and the results are stored line by line in the final database. Well, the elevated number of database calls made the API performance decay, in the long run, hence this proposal was dismantled.

Hereupon, two more similar trials were attempted. In these approaches, the idea of having a staging area persisted along with the dimensions' handling process. The difference between those two new approaches relies on the type of data warehouse and staging area, the first one being a MongoDB database (NoSQL), and the second one a MySQL database (SQL). Compared to the last method presented, the main change underlies the way data are handled during the load of the bridge and facts' tables. Briefly, while in the past the load was based on queries consistent to the staging area, now, before any treatment, the dimensions needed are loaded into dataframes, and those are the structures to be queried along the way.

A full explanation of how all the approaches were developed and how they function can be found in Appendix 2 – Test Cases.

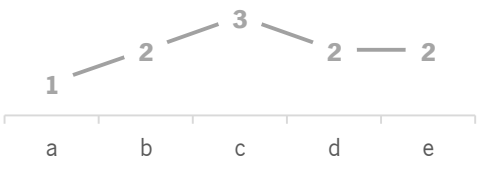
Given this explanation, it is time to consider the points stated at the beginning in order to evaluate the success and applicability of this structure. This can be easily consulted in Table 8, where the first column represents each aspect to be taken into consideration, and the second one the evaluation of those aspects. For example, on the first line, it is possible to consult the "Code Cleanliness" evaluation, in which it is stated how good or bad the different structures are regarding the code's readability and the evolution of this aspect through the different test cases is graphically represented, a comparative scale from 1 to 3 (in which 1 is the minimum and 3 the maximum) having been used for it. The numbers in the horizontal axis represent the different test cases sorted.

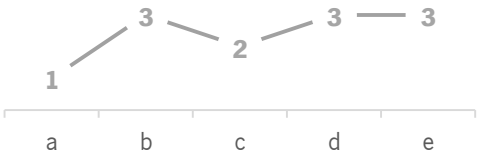
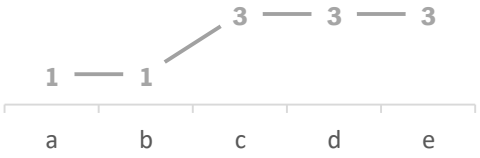
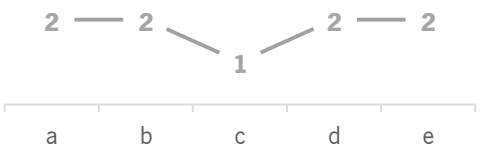
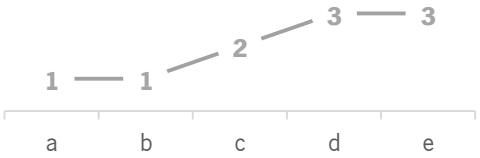
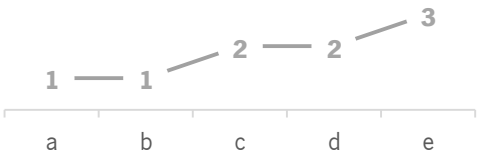
So, to put it in a nutshell, Table 8 includes a graphic for each aspect to be evaluated in which it is possible to see its evolution through the different model attempts (labelled from "a" to "e", according to the label below). Note that the results presented in this table were properly validated by this dissertation's coordinator and his advisor at IOTech, both with experience in this field.

Graphics' Label:

a. Modelling Based on Arrays;
b. Modelling Based on Dataframes;
c. Modelling Based on NoSQL Queries;
d. Mixed Modeling with Arrays, Dataframes and Unstructured Storage;
e. Mixed Modeling with Arrays, Dataframes, and Structured Storage.

Table 8 - Test Cases Evolution.

ASPECT	EVOLUTION
CODE CLEANLINESS	
This aspect corresponds to how easy it is to read a stretch of code, how less complex the system needs to work accordingly. For that, it is understandable that the first attempt is "noisier" once it requires a lot of iterations.	

ASPECT	EVOLUTION
COMPREHENSIVE DATA STRUCTURE This aspect corresponds to the comprehensiveness of the data structures used. For instance, it is easier to understand and retrieve information from data displayed in dataframes than in arrays.	
DATA FIXEDNESS This aspect is related to the persistence of the information into the system. So, it is understandable that only is the cases that make use of staging areas it is possible to achieve that persistence.	
ETL DURATION This aspect is directly related to the time that the ETL process needs to be completed. Has mentioned before, there is no optimal solution for this question, and therefore no top scores.	
SCALABILITY PROCLIVITY This aspect, as the description suggests, is related to the adjustability of the system towards scalable realities. Working with no staging area is a big no for this question, but also executing tons of calls to a database.	
TYPES OF DATA SOURCE This aspect refers to the adaptability of the solution towards different types of data sources (structured, semi-structured, unstructured). So,	

ASPECT	EVOLUTION
it is easy to understand that the last test case is more prepared to deal with this variety, once no matter which type of data source gets in, from there data have a defined structure that does not interfere with the process after the loading phase.	

Looking at the comparison in Table 8, it becomes clear that the last version of the API tested is not perfect but represents good progress in several aspects, the first attempt being the one used throughout this dissertation.

In order to provide some further insights on how the data treatment is handled inside the API, it is possible to see below two complementary sections that lead to the final result: Extract, Transform, and Loading Flow, where it is possible to see, in short, how the high-level data flow occurs, and Generic Treatment, where some verifications that are more generic and can be applied in different contexts are included.

4.1.1. EXTRACT, TRANSFORM, AND LOADING FLOW

As expected, when looking at the data flow, first it is possible to find the dimensions' handling process, then there is the bridge table (that feeds from information already handled in the dimensions), and just afterwards there is place for the facts' table that takes in information prevenient from all other tables. This becomes clear while looking at the code structure presented in Figure 25.

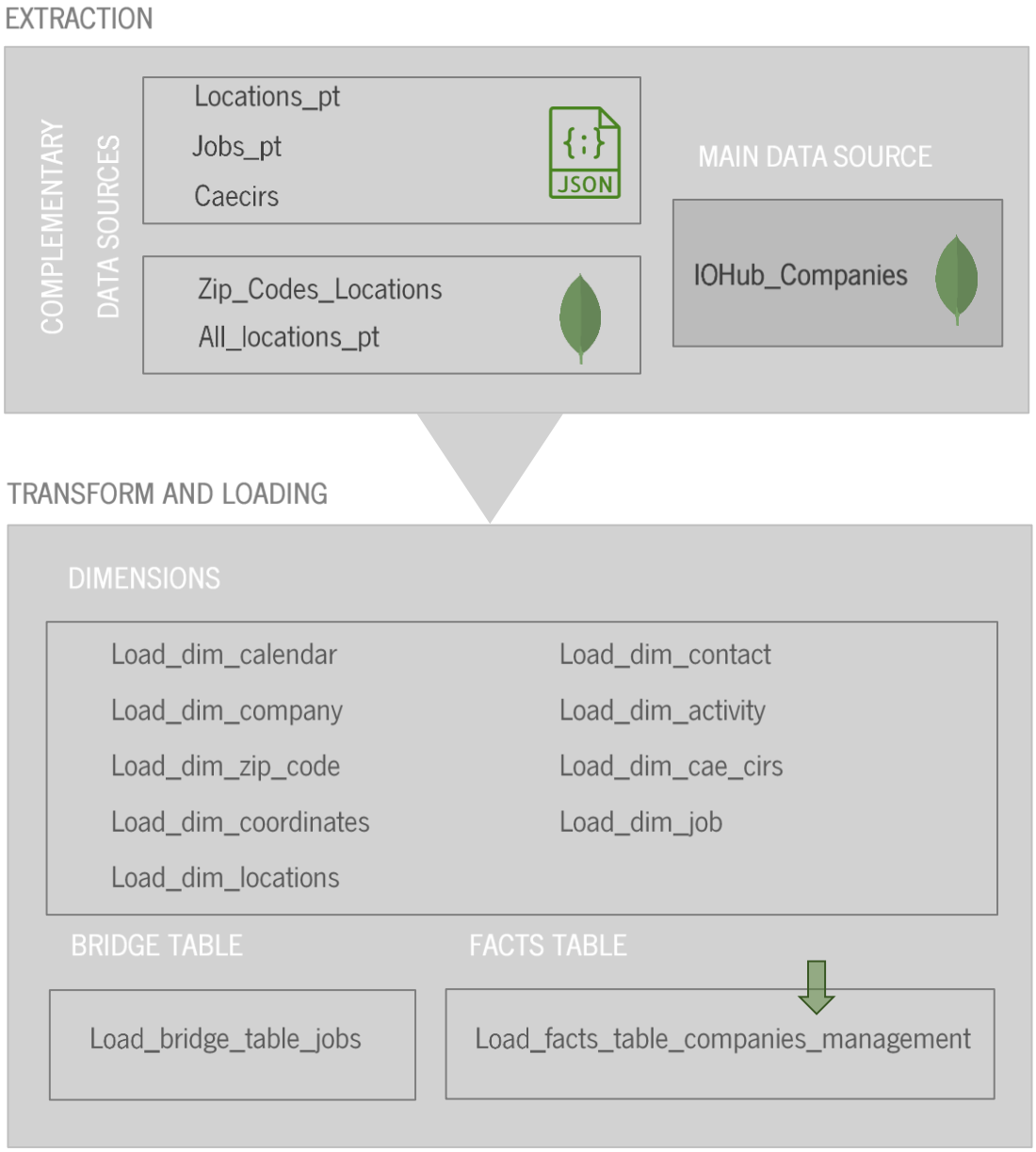


Figure 25 - ETL Flow.

As it is plain to see by looking at the flow presented, in the extraction phase, several data sources are used to fulfill the purpose of standardization and correct the main data as much as possible, transforming it into the most relevant and usable information (further information on these data sources can be found in the case study specifications, Data Sources).

As the main data source the MongoDB collection appears, IOHub_Companies, which is in the base of the case study used during this project. To complement this source or standardize the data stored in it, different sources arise (all other listed in Figure 25).

To retrieve sources that are mongo collections, a connection is established with the database using pymongo library to create a client, where the host, user, password, and database are specified. This client is used to call upon the different collections inside its database, in their specific functions. An example of how this gets done can be seen in Figure 26.

```
def get_Source():
    data = agent.db.source
    query = data.find({})
    data_array = []
    for line in query:
        data_array.append(line)
    return data_array
```

Figure 26 - Example of a mongo collection call.

Regarding the JSON files used, both of them are called using a structure similar to the one presented in Figure 27, but in different moments, for instance, the file zip_codes_locations (in Figure 28 it is possible to see an example of the data in this file) is called when the extraction phase starts, unlike the file all_locations (Figure 29) that is called while treating the data to load the facts' table.

```
with open('source.json') as json_data:
    Data = json.load(json_data)
```

Figure 27 - Example of JSON file call.


```
[{"zip_code_2": "011", "city": "Águeda", "zip_code_1": "3750", "district": "Aveiro"},
{"zip_code_2": "012", "city": "Águeda", "zip_code_1": "3750", "district": "Aveiro"},
{"zip_code_2": "013", "city": "Águeda", "zip_code_1": "3750", "district": "Aveiro"},
...]
```


Figure 28 - Snapshot of the zip_codes_locations file.


```
[{"city": "Águeda", "district": "Aveiro", "parish": "Aguada de Cima"},
{"city": "Águeda", "district": "Aveiro", "parish": "Fermentelos"},
{"city": "Águeda", "district": "Aveiro", "parish": "Macinhata do Vouga"},
{"city": "Águeda", "district": "Aveiro", "parish": "Valongo do Vouga"},
...]
```


Figure 29 - Snapshot of the all_locations file.

The needed functions having been defined, is time to execute the extraction per se by calling them and opening the necessary files. When examining the code this looks like what is shown in Figure 30:

```
companies = functions.get_companies()
locations = functions.get_locations()
jobs = functions.get_jobs()
caecirs = functions.get_carcirs()
parishes_match = pd.read_josn(r'correspondencies.json')
```

Figure 30 - Extraction phase.

Now it is time to start the transformation phase by treating and standardizing the needed data, and then load it into the data warehouse. To facilitate the visualization of this process, this is here presented divided according to the different tables that compose the multidimensional model for the use case under study. In Table 9, a cross-function matrix is presented, making it easier to see which treatments are applied in each table.

Table 9 - Cross-function Matrix.

TABLE	TREATMENTS
Dimension Calendar	No treatments needed, once it is created directly in code.
Dimension Company	In this dimension, there is necessary to validate the NIF or NIPC, in its format and value (Valid_nif). Here is also verified if there is a name associated with the company, otherwise, the company is excluded.
Dimension Zip Code	No treatments needed, once it is populated directly based on uniformized data sources with no error detected.
Dimension Coordinates	This dimension requires validation of all coordinates according to the range in which they must be included (Valid_coordinates, for both latitude and longitude, paired, values).
Dimension Contact	This dimension must see all of its attributes validated accordingly with format and range when needed (Valide_phone (used to verify both phone number and fax number), Valid_site (for the website link structure), Valid_facebook (for the facebook link structure), Valid_email (for the email structure)).
Dimension Activity	No treatments needed, once no errors were discovered regarding its attributes.

TABLE	TREATMENTS
Dimension Cae Cirs	No treatments needed, once it is populated directly based on uniformized data sources with no error detected.
Dimension Job	No treatments needed, once it is populated directly based on uniformized data sources with no error detected.
Dimension Location	No treatments needed, once it is populated directly based on uniformized data sources with no error detected.
Bridge Table Jobs	No treatments needed, once it is populated directly based on uniformized data sources with no error detected.
Facts Table Companies Management	When the time to load this table arises, it is necessary to uniformize all data prevenient from the main data source to match with the one already stored in the different dimensions. Therefore, all the treatments used in the previous uploads are here applied again. But here a few more are required due to error in the main data source data. So, to match zip codes it is used a form and value verification (valid_zip_code) and to match locations it is used a function to uniformize string values (str_format). An advanced check is taken care of while verifying the zip codes which consists of crossing locations and codes to guarantee that the code used actually belongs to the given location (valid_zip_codes_location).

After this quick pick on the treatments handled in each table's processing, it is time to explore each of them and how the sources and treatments appear in their flow a little bit further.

There are some variances in the way the tables are loaded. In order to give a sneak peek on how this is done, two different high level flows are presented: dimensions and facts' table/ bridge table (where it is necessary to compare the values in the dimensions and the ones in sources).

In Figure 31, it is possible to see an example of the flow corresponding to a majority of the dimensions. It is important to bear in mind that, most of the times, the validation of several attributes is required, and sometimes as a group and not individually. It might also include some specific standardizations according to the data sources used.

```
procedure load_dimension ():
    dimension_data = []
    id = 0
    mysql_query = "INSERT INTO dimension (id, attribute) VALUES (%s, %s)"
    cursor = mydb.cursor()
    for line in source:
        if valid_attribute:
            attribute = source['attribute']
        else:
            attribute = null
        end if
        entry = {'id': id, 'attribute': attribute}
        dimension_data.append(entry)
        values = (id, attribute)
        cursor.execute(mysql_query, values)
    end for
    mydb.commit()
end procedure
```

Figure 31 - Dimensions Load Flow.

- **Load_dim_calendar:** this function aims to load the Calendar dimension. This dimension does not require any outside source, the only information that it needs is a beginning date (defined according to the case study's specifications) and the current date, from there all dates that compose the dimension are generated inside this function. The fact that this is a "data creation" function makes any date verifications unnecessary. All necessary creations result in sending the final data to the database, supporting the data warehouse, more specifically, the table Dim_Calendar.

- **Load_dim_company:** this function's goal is to load the Company dimension into the data warehouse. Unlike the previous one, this function requires a data source, the main one, IOHub_Companies. Given the time of information going through this dimension, the only generic treatment function required is the valid_nif to decide which values get to be inserted in the data warehouse table. All necessary transformations resulted in sending the final data to the database supporting the data warehouse, more specifically to the table Dim_Company.

- **Load_dim_zip_code:** this is the section destined to load the Zip Code dimension. To do so, a complementary data source was used, zip_codes_locations. Once this source is reliable and it includes all the Portuguese zip codes, there is no need to perform any treatments or

validations. This function also includes the creation of a so-called zip complete attribute, that is nothing more than the result of combining both parts of a zip code in one attribute. All the necessary transformations resulted in sending the final data to the database supporting the data warehouse, more specifically to the table Dim_Zip_Code.

- **Load_dim_coordinates:** this function's purpose is to load the Coordinates dimension. As its input, this function requires only the main data source, IOHub Companies, from where all paired latitudes and longitudes are retrieved. To check this information, so as to avoid inserting it wrongly into the data warehouse, the function valid coordinates is used beforehand to lead the treatment process. All necessary transformations resulted in sending the final data to the database supporting the data warehouse, more specifically to the table Dim_Coordinates.

- **Load_dim_contact:** this function aims to load the data warehouse table that supports the Contacts dimension. This function's unique input is the main data source, IOHub Companies, from where all attributes related to the companies' contacts are retrieved. This function involves a lot of uncertain data. Therefore, it requires more verification than the dimensions presented above. Bearing this in mind, this section of the transformation phase supports its verification on the valid email function, to check the email and the phone number's validity, to check both fax and phone number's format, website validity, the link's structure, and to check Facebook's validity and link structure. All necessary transformations resulted in sending the final data to the database supporting the data warehouse, more specifically to the table Dim_Contact.

- **Load_dim_activity:** this function is used to load the data associated with the Activity dimension. Once again, the main data source is here called to provide the data needed to fulfill the attributes inside this dimension. This particular set of data does not imply any specific treatment function since no errors are involved, just null lines to be handled inline. All necessary transformations resulted in sending the final data to the database supporting the data warehouse, more specifically to the table Dim_Activity.

- **Load_dim_cae_cirs:** here the necessary transformations to load the Cae Cirs dimension are presented. This function receives a complementary data source, cae_cirs, as data input, which allows completing this table in the most accurate and complete way. Once the source used is

a reliable source and does not present errors and no standardizations are to be done, it is not necessary to call any treatment function here. All necessary transformations resulted in sending the final data to the database supporting the data warehouse, more specifically to the table Dim_Cae_Cirs.

- **Load_dim_job:** this function is associated with the Job dimension's transformation and loading. In order to fill this table with all correct and valid options possible, a complementary data source is used, jobs_pt. Since the data source is considered a reliable source, there are no transformations to be done here. All necessary transformations resulted in sending the final data to the database supporting the data warehouse, more specifically to the table Dim_Job.

- **Load_dim_location:** this function includes the processes required to fill the table correspondent to the Location dimension. This function receives several data sources that complete themselves and allow to create a complete background to fill up this dimension. Those different data sources are the same used to mount the complementary data source zip_codes_locations (cities and districts) and the complementary data source all_locations. The way this process is mounted implies that to have the attribute "parish" filled, a line must have both city and district attributes filled (and so on). This helps to reduce ambiguous cases in which there is a parish name that exists in different cities/districts, and therefore, represents irrelevant information. All necessary transformations resulted in sending the final data to the database supporting the data warehouse, more specifically to the table Dim_Location.

In Figure 32, it is possible to see an example of the flow correspondent to both bridge and facts' tables. It is possible to see that this time a source and a dimension are called to be compared in order to retrieve the dimension's needed id. Furthermore, and particularly in the case of the facts' table, there is a restriction: the facts to be included must be not null.

```
procedure load_bridge_facts_tables ():
    table_data = []
    dimension = get_dimension()
    mysql_query = "INSERT INTO dimension (id, fact) VALUES (%s, %s)"
    cursor = mydb.cursor()
    for line in source:
        if fact not null:
            if valid_attribute:
                attribute = source['attribute']
            else:
                attribute = null
            end if
            line = dimension.loc[(dimension['attribute_dimension'].values == attribute)]
            id = int(line['id_dimension'].values)
            entry = {'id': id, 'fact': fact}
            table_data.append(entry)
            values = (id, fact)
            cursor.execute(mysql_query, values)
        end if
    end for
    mydb.commit()
end procedure
```

Figure 32 - Bridge and Facts Table Load Flow.

- **Load_bridge_table_jobs:** this section includes the process necessary to fill the bridge table Jobs along with a complimentary table Group Jobs (used uniquely to avoid multiple connection database restrictions). Here both the main source, IOHub Companies (to know which jobs belong to which groups), and the Jobs dimension already in the data warehouse (to provide the ids needed), are used as input. This process does not require any treatment function since the process here is the establishment of a correspondence and not a transformation process per se. All necessary correspondences performed result in sending the final data to the database supporting the data warehouse, more specifically to the table Bridge_Table_Jobs, along with the support table Dim_Group_Jobs.

- **Load_facts_table_companies_management:** now that all the dimensions and the bridge table are loaded, it is time to load the Companies Management facts' table, possible through this function. Before taking care of the necessary transformations, all dimensions and the bridge table are called from the data warehouse (done as shown in Figure 33).

```
dim_company = functions.get_dim_companies()
dim_location = functions.get_dim_locations()
dim_jobs = functions.get_dim_jobs()
dim_caecirs = functions.get_dim_carcirs()
dim_calendar = functions.get_dim_calendar()
dim_coordinates = functions.get_dim_coordinates()
dim_activity = functions.get_dim_activity()
dim_zip_code = functions.get_dim_zip_code()
dim_contact = function.get_dim_contact()

bridge_table_jobs = function.get_bridge_table_group_jobs(dim_jobs)

load_facts_table(dim_company, dim_location, dim_caecirs, dim_calendar, dim_coordinates,
dim_activity, dim_zip_code, dim_contact)
```

Figure 33 - Dimensions and Bridge Table call from the Data Warehouse.

To retrieve the necessary ids from their dimensions, and from the bridge table, a direct match is required by comparing the value received from the main source with the values inside those dimensions. It is understandable that some standardizations are needed, once here matches with data provided by different sources are being done.

To start this process, all attributes, along with all facts are called to be compared with the dimensions. These facts are first verified to check if there are no null values, otherwise, the line is discarded.

The first dimension to be taken care of is the **Company dimension**. Here it is only necessary is to validate the NIF and form a line with the data acquired to compare it directly using the company name along the NIF.

Next to it, to link this table, the date underlying the object in the database is retrieved and compared to the complete date in the **calendar dimension**.

Then, it is time to take care of the **coordinates dimension**. To standardize the data, the valid coordinates function is called again and after its application both latitude and longitude are compared, as a pair, with the data inside the dimension.

After that, the attributes of the contact are run through their validation functions (validate email, website, phone number, and Facebook) accordingly, and compared as a group to the data inside the **Contacts dimension**.

The following dimensions are the ones storing data from the **activities and cae cirs**. These two dimensions are handled in a similar way: the attributes needed are called directly from the main data source and compared to the ones in each table. In the first case, both activity and category are used, and in the second one, only the cae cirs code is used.

The **Location dimension** is the next in line, and probably the one requiring the greatest effort to retrieve the correct id from. Here, the first thing to be done was the creation of a dataframe containing only the islands' information, and another one with all locations after having been run through the str_format function (standardizing all names). That having been done, it is time to look at the data that is coming from the main data source.

In a lot of cases, not a name, but an ObjectId from the complementary data source locations_pt in MongoDB is coming from the main data source. Therefore, a verification was developed to reach the true names underneath those ids.

After that, the first verification regarding the connection is about whether the data received correspond to an island or not. If so, the city is checked to avoid having district names there, which in this case means the archipelagos.

Then all lines are run through a similar set of verification as well the load dimension to verify both district and city. The differences start when the time for parishes comes. This major work is related to a reform that takes place in 2013 in Portugal in matters of the union between parishes. To deal with this ambiguous situation, a set of steps to go through were defined:

1. Direct comparison between the parish (always having the city and district as a base) and the list in the database;
2. Compare the different words that compose the parish's name and associate the one with a higher number of words in common;
3. Retrieve the correspondent parish's name before the 2013 reform and establish a direct comparison;
4. Compare the different words that compose the parish's name before the 2013 reform and associate the one with a higher number of words in common.

If after all these attempts there is no connection, then the parish's name is considered null.

Then, it is time to establish a connection with the **Zip Code dimension**. Similarly to the dimension process, the Zip Code is validated, and the complete zip code is formed and used to realize the comparison.

Last but not least, the **bridge table's** connection is taken care of. The first thing to be performed here is to put the jobs' array received into a simple list of ids that can be easily compared with all the lists stored in the bridge table.

All necessary transformations and correspondences established resulted in sending the final data to the database supporting the data warehouse, more specifically to the table Facts_Table_Companies_Management.

4.1.2. GENERAL TREATMENT

Here the more generic functions created and used to treat and validate the information underlying this project are presented. In order to have a better understanding of their applicability, a brief explanation is given on each and every one of them.

- **Is_leap_year**: as the name suggests, this function is about verifying if a given year is a leap year or not in order to check if the days in February are correct or not in further analysis. For that to become possible the following rule was used: the given year must be a multiple of four (4), of one hundred (100), and of four hundred (400).

- **Month_days**: this function has the purpose of verifying if the number of days in each month is correct, considering all valid inputs as months' identifiers. This function counts with the support of the previous one to accomplish its mission.

- **Str_format**: this function aims to standardize all strings according to the UTF-8 charset and to ignore some specific words in the parishes' names in order to make it possible to match different data sources at this level.
  For example, if the parish "União de Freguesias de Nogueira do Cravo e Pindelo" is submitted to this treatment, the output would be "nogueira cravo pindelo", allowing the comparison to occur based on keywords.

- **Valid_coordinates**: this function serves the purpose of verifying if a given latitude and longitude represent a valid map point, in other words, if the values of those two fields belong to the range [-180, 180]. After the first verification it is time to check, considering the case study being examined, if the values' input, more than valid, corresponds to a point inside Portugal, either continental or in the islands. To achieve this goal, a rectangle

strategy was used, this means that extreme points of each area were introduced to serve as a reference to the boundaries.

- **Valid_nif**: this function serves to check a NIF ("Número de Identificação Fiscal", which means Tax Identification Number). To reach that goal, the verification rules were first studied and then translated into the system. The basic rules are:
  a. The number must contain nine digits;
  b. The first digit must be 1 or 2 if the number corresponds to a singular entity, or 5, 6, 8, or 9 if the number corresponds to a collective entity;
  c. The last digit is called the control digit and has a specific form underlying its creation: 9*d1 + 8*d2 + 7*d3 + 6*d4 + 5*d5 + 4*d6 + 3*d7 + 2*d8 + 1*d9 (d1 to d9 are the different digits that compose the NIF in an orderly manner, for example, d1 is the first digit. The result of that form's application must result in one of two options so the NIF can be valid:
     i. The result is a multiple of eleven (the remainder of the division is zero (0)), and the NIF is automatically considered valid;
     ii. The result is not a multiple of eleven (11), but when divided by it the remainder is one (1). If allied to it the last digit is a zero (0) another verification is done: ten (10) is added to the result of the form's application and then the result is divided by eleven (11); if this new number reveals itself as a multiple of eleven, then the NIF can be considered valid.

- **Valid_phone**: this function name might be misleading, once it does not allow the verification of phone number's validity, as, for instance "Is this number actually assigned to someone?" but only "Is the format of this number correct and possible?". Bearing this in mind, and considering the context of the case study, the given number must contain either nine or thirteen characters. If it has nine characters that are digits, then the number is automatically considered valid. Otherwise, if it has thirteen characters, at least twelve of them (excluding the first character) must be digits, and the first one must either be zero (0) or a plus sign (+), so this number can be considered valid.

- **Valid_site**: this function, similarly to the previous one, does not validate a given link if it leads to a valid website, but only if the link has a proper structure. For that to be possible, some rules were defined such as: the link must contain at least two components (it is considered a component each part of a link separated by a dot; for example, www.google.com is a link that contains three components "www", "google", and "com"), and the identification of the protocol (http and https) along with the form "www" must not be considered mandatory for the link to be valid (once it is possible to reach the website without inserting those specifications while browsing it).

- **Valid_email**: this function is mounted in a base similar to the valid_site one, considering the email's components, among other rules:
  a. The given email must contain at least one dot and one at (@);
  b. The components are achieved by being split, initially, by the "at" sign and then by a dot. The total of number of components must be, at least, three, the component before the "at" (the user name), plus two other components next to the domain, the hostname and the top-level domain (company or country identifier). For example, in the email "gisela@gmail.com", there are three components: "gisela", as the user name, "gmail", as the hostname, and "com" as the top-level domain;
  c. Each component must not be null or blank.

- **Valid_facebook**: similarly to the website and email validations, the Facebook link validation is made based on its components. To be considered valid, the link must contain two components, the first one being either "www.facebook.com/" or "https://www.facebook.com/" or even "http://www.facebook.com/".

- **Valid_zip_code**: this function serves as a support to the valid_zip_codes_location function by being a first verification of the zip codes' validity. For that, the context of the case study is substantial, once the zip codes' structure changes significantly from country to country. So, to validate the zip code, a complementary data source was used. This source is composed of all zip codes, both the first and second parts of them, existing in Portugal. A given zip code gets approved only if it exists in the complementary source.

- **Valid_zip_codes_location**: this second verification is not meant to check the validity of a zip code, but its concordance with the given location paired with it (just the city and the district are considered here). To do so, the same data source as before is used, and if all given data (both zip code parts, city, and district) match a line in the data source, then the zip code is considered as correctly paired with the location.

Right now, every single time that the user wants this process to be applied, he / she must manually clean the data warehouse and then execute the process. Well, this is not ideal, a proper solution must have some mechanism that executes this process from time to time, a Refresh System (3). Unfortunately, it was not possible to develop such a mechanism during this dissertation, but it must be dealt with in the future.

When dealing with high levels of data volume, cleaning all data warehouse to reload might not be the best approach. Instead, the idea beyond this mechanism is to acknowledge if there were any changes in data or any new entries, and where that occurred, in order to update only the necessary tables and rows. For example, to verify new entries, a support table must be created to indicate the last update on each table and rows' count.

The RESTful API work having been concluded, the data are loaded to a MySQL database (1.4), as mentioned, this one being the data warehouse (identified on the architecture) used to support the next architecture phases. The multidimensional model of the case study used can be consulted ahead in the section Case Study: IOHub Companies.

4.2. ANALYSIS DEVELOPMENT LAYER

All processes needed to handle the data are included on this layer, the data currently stored in the data warehouse, and produce significant analysis over it. In Figure 34, it is possible to verify that this development layer includes both Data Warehouse and OLAP layers from the high-level architecture. This is the phase in which the main innovation of this project relies, making it a pervasive solution.

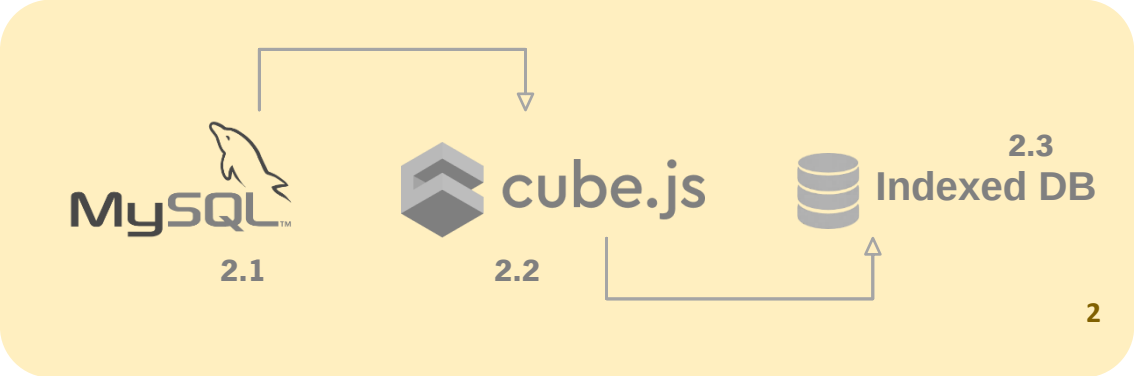


Figure 34 - Analysis Development Layer.

The main work here is directly related to the OLAP layer in the architecture, which relies on Cube.js (2.2). Basically, a cube structure is defined here in order to allow the aggregation function over data, along with roll-up, drill-down, and slice and dice functions over the data stored in the data warehouse (2.1).

Considering the nature of the multidimensional model defined for this project, the creation of a polymorphic cube was required. A polymorphic cube, considering Cube.js' documentation, is a cube composed and fed by more than one table. To build this type of cube, several simple cubes must be built, and then their relations must be established. The "individual" cube, corresponding to the facts' table, must join all keys in order to allow the flexibility and cross-querying implied in the cubes.

For example, every dimension's structure must be loaded as presented in Figure 35, in other words, each dimension must have a particular cube, with a call to its corresponding one in the data warehouse, and all attributes in the database assume a dimension role inside this type of cube.

```
cube ('dimension', {
    sql: 'SELECT * FROM database.dimension',
    dimensions: {
        id: {
            sql: '${Dimension}.id',
            type: 'number',
            format: 'id',
            primarykey: true
        },
        attribute: {
        sql: '${Dimension}.attribute',
        type: type
        }
    }
})
```

Figure 35 - Simple cube structure for a dimension.

Regarding the facts' table's cube, it must be loaded as in the example presented in Figure 36. This means that, as in the dimensions' cubes' case, this cube must contain the connection to the facts' table in the data warehouse, but now, in the dimensions' section, instead of having attributes, the primary keys of all cubes that represent dimensions and must establish connection with the facts' table being studied are included. These keys are directly retrieved from the other cubes, but for that to be possible, another section is needed, the joins section. In the joins section, the type of relationship (for this case it must be "belongsTo", once the goal is to retrieve attributes from another cube) is specified and the representative attributes are linked with the original ones in their main cubes. This cube has another section, called measures, where all facts are identified and retrieved directly from the data warehouse.

```
cube ('Facts, {
    sql: 'SELECT * FROM database.facts_table,

    joins: {
        Dimension: {
            relationship: 'belongsTo',
            sql: '${Facts.idDimension} = $ {Dimension}.id_dimension'
    },
    dimensions: {
        idDimension: {
            sql: '${Facts}.id_dimension',
            type: 'number',
            format: 'id',
            primarykey: true
    },
    measures: {
        sql: '${Facts}.count,
        type: 'count'
    }
})
```

Figure 36 - Simple cube structure for facts' table.

The bridge table schema is defined in a similar way to the facts' table. The main difference is the inexistence of a measures' section.

Once the multidimensional model has a star structure, and, therefore, has only one facts' table, as well as the required analysis, it was concluded that there is no need to develop more than one cube. This conclusion results in the cube built presenting the same structure as the data warehouse, as it can be seen in Figure 37.

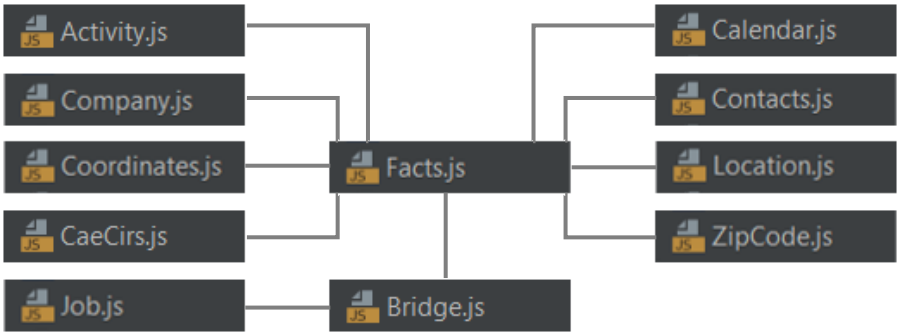


Figure 37 - Cube schema (Case Study: IOHub Companies).

The cube's schema definition having been determined, it is time to make the most of the data underlying it. For that to become a reality there were two possibilities, the first one involved querying Cube.js directly at all times when a different query is needed, the second one involved loading some base data into a cashing system, IndexedDB (2.3), and work based on the data available there, and query Cube.js only when new information is needed. The path chosen to deal with these matters was the second one, and it is further explained in the following subsection.

4.3. CACHING DEVELOPMENT LAYER

This layer includes all processes needed to handle the querying mechanism over data and how it flows inside the system. In Figure 38, it is possible to see that this development layer covers both OLAP and Visualization layers from the high-level architecture. Nevertheless, this mechanism is not represented by a different stage in the architecture since it is a sublayer of the two main layers mentioned here.

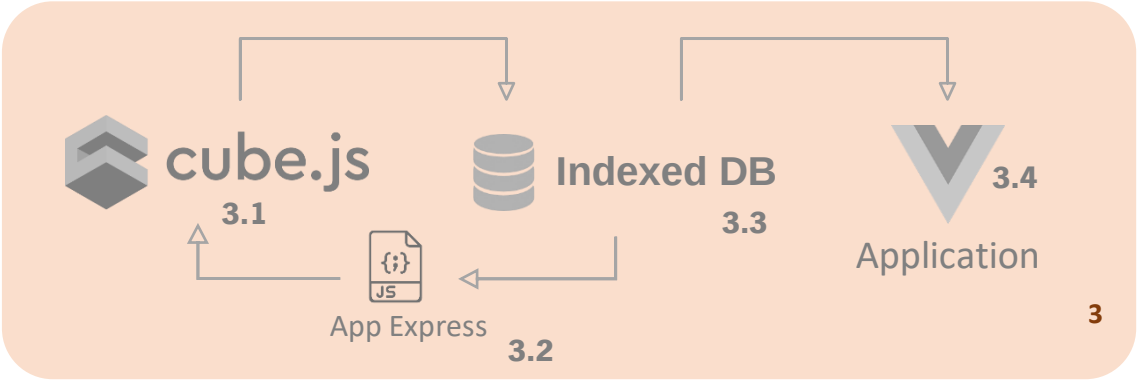


Figure 38 - Caching Development Layer.

In order to get a better understanding of how the connection between the OLAP Layer (3.1) and the User Interface (3.4) is developed, a simple flow schema identifying the processes involved is presented in Figure 39.

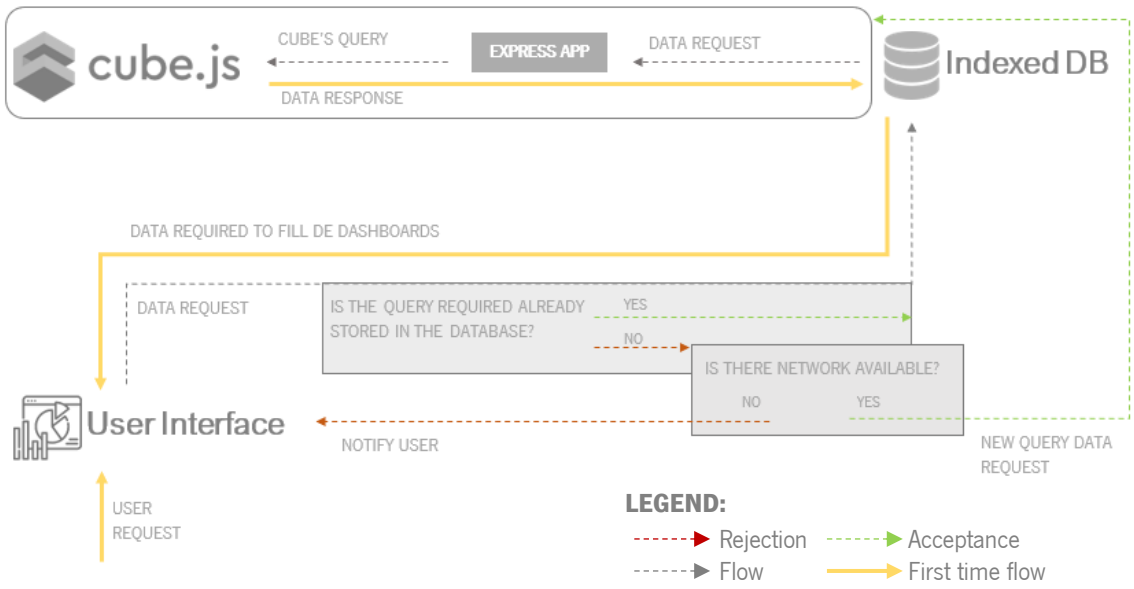


Figure 39 - Caching and querying mechanism.

In an initial phase, the process starts in the intermediate database, IndexedDB, which is initiated with a set of object stores, one per each of the basic queries required for the predefined analysis, and another one in which the name of all other object stores and the query used to fill them up are stored. The creation of the database having been concluded (as shown in Figure 40), a set of requests are made based on the queries required to fulfill the charts presented whenever the application is opened. This means that the set of data requested is presented in its pure state, without any filters applied. One example is presented in Figure 41. As it can be seen there, this process is taken in two separated functions, one to request the data from Cube.js (for example, readQuery), and another to insert the data into the designated table with the specified key for the table (for example, loadQuery).

```
procedure db.onupgradeneeded = function (event) {
    var database = event.target.result
    var query = database.createobjectstore('table', {keypath: 'key'})
    query.createindex('key', 'key', {unique:true})
}
end procedure
```

Figure 40 - IndexedDB creation.

```
procedure loadQuery (db) {
    readQuery ( function (data) {
        var tx = db.transaction('query', 'readwrite')
        var store = tx.objectstore('query')
        data.foreach( function (x) {
            var id = x['id']
            store.put([key: id, value: x})
        })
        tx.oncomplete = function () {
        db.close()
        }
    })
}
end procedure

procedure readQuery (callback) {
    $.get('route/query', function (data){ callback (data) })
}
end procedure
```

Figure 41 - Example of a basic query request.

For that to happen, it was necessary to develop a simple express application (Express App (3.2)) which receives a route request in firsthand (from IndexedDB (3.3)) and then requests the query, implied to that route, to Cube.js. Once the request is fulfilled, the data acquired from the OLAP layer is sent to IndexedDB, and the response to each request is allocated in the corresponding object-store. This database loaded, the system has now data available to initialize the set of dashboards included in the application.

It is important to bear in mind that, for the first time, the application is open; it is mandatory to have an internet connection, otherwise it is not possible for IndexedDB to reach the OLAP layer, neither for Cube.js to access the data warehouse. After that, the user can use the application

without connecting to the internet, since the data available in IndexedDB is used as long as the same browser session is used.

1. The user can apply filters over the data retrieved based on the basic queries. Well, those filters imply querying IndexedDB and, possibly, Cube.js about the new request data. So, in order to succeed in in this task, as soon as the user interface receives a user request, it must be processed in the following way: first, the request is sent directly to IndexedDB, and some verifications are made along the way:

- Is the required query stored in the database? In other words, has this query been ever asked for?

Well, if the query has, indeed, been asked after the browser session has been started, the information related to it should already be stored in IndexedDB and, therefore, the process ends with a response from IndexedDB displaying the data requested.

In the second case scenario, if the query has never been asked since the beginning of the browser session, the process gets a little bit longer, for it implies querying the OLAP layer. For that to happen, a stable network connection that leads to the next verification is obviously necessary.

- Is there a network available?

If the conclusion of this verification is that there is no network available, the process is concluded with a notification sent to the user, letting him know that for this process, an internet connection is required.

In case of a positive response, the query is sent to the express application that is ready to receive any valid query and send it to Cube.js. Once the query has been received, the data are set and sent, first, to IndexedDB, where a new object store is created to hold the data related to the query submitted, and the query is stored in the queries' object store (for further verifications). When all data are properly stored in the database, the charts are updated with the new information provided by IndexedDB.

4.4. VISUALIZATION DEVELOPMENT LAYER

This last layer includes the means needed to present the system's results to the users. In Figure 42, it is possible to verify that this development layer has a direct connection with the Visualization layer of the high-level architecture.

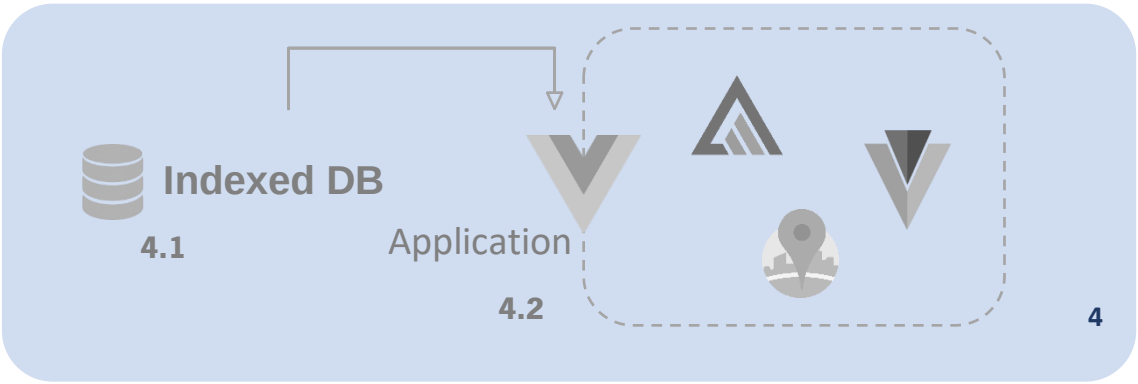


Figure 42 - Visualization Development Layer.

In short, this development layer consists of consuming the data stored in the IndexedDB database (4.1) and displaying it through relevant charts and dashboards integrated in a user-friendly application (4.2).

To do so, different technologies were merged to suit different purposes of an integrated reality of which this system is the endpoint. In the base of the application's creation a JavaScript framework was used, Vue.js, that was complemented with some libraries such as Apexcharts, used to build almost every single chart presented in the platform, and Google Maps API to build map dashboards. Another complement added here is Vuetify, to create different and more appealing components; this appeared because the template downloaded to serve as a base to the application construction had some dependencies on it.

4.4.1. STRUCTURE CODE COMPONENTS

This subsection is intended to present how the application is structured and how the different code sections are distributed through the project, in other words, the folders' structure used and a global overview on the solution.

So, it is important to mention that to help with the application's aesthetics, a Vue.js template (Vue Material Admin Template) was downloaded; that came with a predefined structure that, with some adjustments, originated the application's structure (that is further explained in the section Application Structure). Therefore, the project includes a set of folders with predefined structures (such as charts, cards, tables, etc.) and definitions.

Over that, different structures were included to suit the project purposes, either functional or visual. Below, it is possible to check how these components are grouped inside the project according to their purposes:

[1] **Public**: inside this folder are stored a set of static assets, such as the logos;

[2] **Src**: this folder is divided into some other ones in order to organize the code the best way possible:

a. **Apexcharts**: as the name suggests this folder is the one containing the Vue components relative to the charts used based on the Apexcharts' library;

b. **Components**: this folder was already created in the downloaded template and contains a set of predefined components such as tables, charts, cards, forms, etc. From these components, only two tables and the structure of a card is used through the dashboards included in this project

c. **Router**: this folder is the one containing the needed files to configure the different routes inside the application;

d. **Views**: last but not the least, Views folder is the one containing the different views structure, header, side menu, and so on;

Also in Src, some other relevant documents can be found, such as:

e. **IndexedDB**: this is the file responsible for the creation of the browser side database and its fulfilment. Furthermore, some functions to handle data filters are here included;

f. **DatabaseRequests**: this file includes a set of functions used to retrieve the data from the IndexedDB database and fill up the different dashboards' components;

g. **GeneralFunctions**: this file is the one containing a group of generic functions used through different components, like a function to create queries' names when a filter is applied;

h. **App**: It is also here that the application motor is located initiating all needed processes to the application run properly.

4.4.2. APPLICATION STRUCTURE AND DASHBOARDS

In this section the set of dashboards created to suit the project's business analytical needs and the application's structure are explained.

The application is structured according to the dashboards that compose it. Basically, from the header, it is possible to have access to the side menu (visible in Figure 43) where four links are included, each one of them allowing the access the different dashboards.

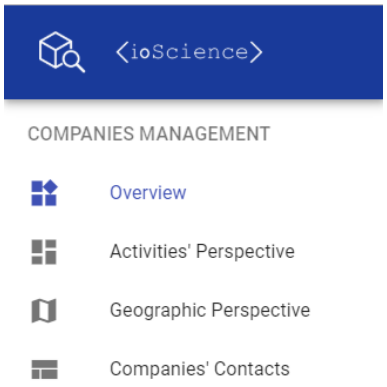


Figure 43 - Side Menu Structure.

In the context of this dissertation four different dashboards were developed aiming to answer different questions, each of them accessible by the following link in the side menu:

- **Overview**: this is the first dashboard to be displayed when the application is opened and it is the one with the most generic analytical questions to be answered. So, it is possible to check a group of cards with global statistics in it, such as the number of companies, a perspective over the companies that are providers or not, either in general, or crossing it over the top ten activities. With this dashboard, the goal is to obtain a global perspective over the countable meaning and then analyze which type of companies are the most relevant for the case study (providers or non-providers).

- **Companies Contacts'**: this dashboard is composed of a single table that allows cross-search through/filtering the companies' contacts, along with sorting their data through different columns. There is no proper analytical question to be analyzed here, its mere function is to display useful information on the companies in an easy way so that the user can find each company based on different attributes, increasing the solution's flexibility.

- **Activities' Perspective**: the goal of this dashboard is to analyze the companies' activities' context in matters of the different attributes related to it. So, this is composed of an analysis considering the CAE CIRS, the activities themselves, their categories, and the groups of jobs associated with them. All charts included here are organized according to each attribute's top ten, individually, and these can be filtered according to the other activities' attributes.

- **Geographic Perspective**: with this dashboard, it is possible to get a geographic perspective on the companies' impact. In other words, it is possible to check the geographic distribution of the companies, and where they are most concentrated, along with an analysis of this distribution crossed with the top ten activities grouped by district. If a filter by district is applied, then the information is grouped by each city belonging to the chosen district.

In Figure 44 and Figure 45, it is possible to see some components integrated across the different presented dashboards. The components 2 and 3 are in the group of high-level analysis, component 1 corresponds to the contacts' list (where the information is blurred due to confidentiality matters), and all other components presented here are directed towards an analysis of the activity. The components 5 and 6 are connected, once the data table is filled according to the data in the pie chart. It is possible to see that, for example, the group of jobs number 88 sees itself represented 3037 across the companies stored in the system; in table 7, it is possible to check which jobs are included in that group.

Figure 44 - Examples of Generic Dashboards' Components.

Figure 45 - Examples of Subject Directed Dashboards' Components.

5. CASE STUDY: IOHUB COMPANIES

This section presents the work developed according to the "IOHub Companies" case study. For that to be fully understood and documented, first a business comprehension subsection is presented, in which included where, who will use the solution, and what for have been. Then all data sources used, main and complementary ones, are presented, followed by data analysis to understand which errors and gaps exist in data to be corrected or filled in order to make the data more accurate and usable. Straightaway, the multidimensional model is included, along with a description of each dimension, fact's table and bridge table.

The topics presented here are considered as a first approach to the data used to test the system developed, so as to proceed fully conscious of what to handle in matters of treatment and standardization on the RESTful API phase of the architecture, conscious of in which direction to turn the analysis to be done, and of which dashboards must be linked to them.

Before going further on this case study, it is relevant to introduce what the IOHub, from where the main data source is retrieved, is. So, IOHub is a collaborative and intelligent cloud-based platform, accessible anywhere, anytime and in any device, and it contributes to the quick and effective resolution of various citizen concerns in several areas, and simplifies how the organizations interact with their clients. From the providers' point of view, it aims to provide a real-time response to citizens' concerns without bureaucracy, and the dissemination and recommendation of services by providers with weak capabilities to spread their work.

In short, IOHub is a platform developed by IOTech used as proof of concept of the IO paradigm (Innovation on), that aims to facilitate the contact between services' providers and the citizens, being that for mere consultation of the services' availability, for help requests, or clarification of any questions.

In order to have a better understanding of the workflow underlying this project, an integrated schema can be checked in Figure 46. It is possible to see a series of steps that are taken care of before the actual developments begin. So, the schema is structured in four flow levels:

- **Business Comprehension Level Flow**: this first level, regarding all the other ones, is not directly connected to handling the data but with a business side approach to the problem. At this level, in a first instance, it is necessary to understand the business context and the problem's context (this approach was taken care of at the beginning of

the project and can be checked in the first chapter of this dissertation); from there it becomes important to know the recipients of the final output, who will actually use this, to build the analysis according to what might become handy to them. Knowing the users of the system, it is time to define a set of KPIs to ensure the reporting phase will include relevant and appropriate dashboards and not just a set of indicators that do not allow to retrieve relevant conclusions (this can be explored in the Business Comprehension section).

- **Data Preparation Level Flow**: at this level, all the work related to preparing the data that will support this project is taken care of. For that, in a first instance, it is essential to gather proper and relevant data sources (which can be seen in the Data Sources section), from there a complete and careful analysis must be handled in order to meet in time any possible errors, the missed values, and the fields that might need some standardization, when the time to cross the different sources comes. Furthermore, a full comprehension on which fields are included in the data sources and have an interest to the case study (this is presented in the Data Analysis section) is required. These two tasks having been concluded, it is time to design the multidimensional model that is a representation of this case study through the whole project (this can be further explored in the Multidimensional Model section).
- **Data ETL Level Flow:** this is composed by the first three components of the high-level architecture, and how its name suggests, this level takes care of what the data processing is, from its extraction until its loading into the Data Warehouse involved in this project. Further explanations on each component can be found in the previous section, Solution Development Layers.
- **Data Presentation Level Flow**: similarly to the Data ETL level, this level is composed of the last two high-level architecture components and the cashing development layer. This level includes all the processes needed to build flexible and extensive analysis through data. More details can be found in the Solution Development Layer section.

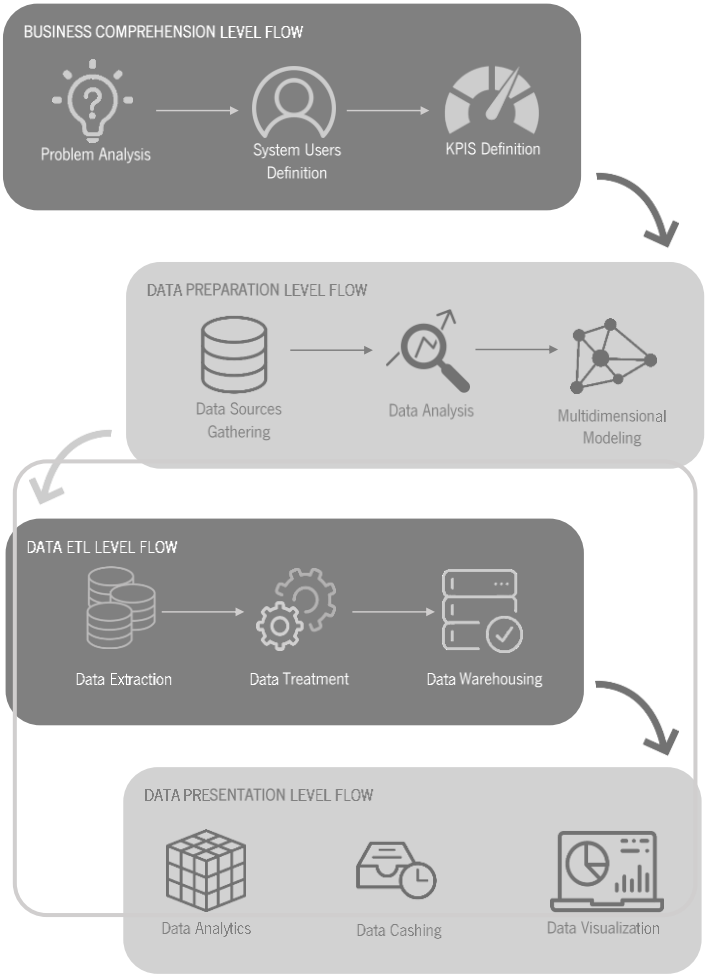


Figure 46 - Solution WorkFlow.

5.1. BUSINESS COMPREHENSION

This case study is fully supported in the IOTech context, the data providers being essential for this work to succeed. This case aims at answering questions related to the companies' management inside IOTech platforms, for example, ioHub.

5.1.1. SYSTEM USERS

While building up a solution that ultimately supports the decision-making process by displaying a set of relevant information, it is quite important to understand who has access to this platform or parts of it, for two different reasons: data access restrictions might be involved, and not all information might concern the same user groups. Therefore, this process starts by identifying who gains access to the information and whether there are different groups of users.

In this particular case study, only one kind of user was identified: the enterprise's administrative layer. Given that, different access dashboards are not needed, once all people involved in the administrative layer can have access to all of the information displayed in the dashboards.

5.1.2. KEY PERFORMANCE INDICATORS (KPIs)

In order to redirect the business study in an efficient way, it is extremely important to find out which information is the most relevant in the decision-making process. For that, it is useful to identify a limited group of KPIs, Key Performance Indicators, based on the goals that the company aims to achieve. Table 10 allows picturing this process by transposing how the KPI is obtained. So, first of all, the business goals are identified (presented in the first column), then it is time to identify, within those business goals, which Critical Success Factors (CSF) are involved (presented in the second column). But how are the CSFs measured? That is why measures must be defined, playing the valuable and easy quantifiable role associated with CSFs (presented in the third column). Finally, the KPIs are presented (in the fourth column), ultimately materializing the goals in one number which allows identifying a positive, negative or stagnant result.

Table 10 - Key Performance Indicators.

BUSINESS GOALS	CSF	MEASURES	KPI
Increase the number of companies involved	Increase the number of companies registered	An increasing number of companies who are registered into the system	Number of companies registered per month
	Increase the number of companies owning an account	An increasing number of companies who have an account in the system	Number of companies with account per month
Reduce the number of companies who leave the system	Decrease the number of companies who get inactive	The decreasing number of companies who are inactive into the system	Number of companies inactive per month
Improve the quality of the relationships	Increase the number of companies rated with four or more stars	An increasing number of companies rated with more than four stars into the system	Number of companies with at least four stars per year

BUSINESS GOALS	CSF	MEASURES	KPI
	Improve the feedback given to the companies	The increasing number of votes per company	Number of votes per company

It is important to bear in mind that this is a first approach to the case study under scrutiny and much more can be done once the platform starts being used by a larger group of users, and more data get inserted through it. The platform used here is fully prepared for this evolution, and its scalability is simple, allowing many more indicators to be defined.

5.2. DATA SOURCES

In order to fulfill the requirements of the present task, it has become necessary to use more than one data source. Therefore, in this process, five complementary sources are used to increase the data quality of the main one.

So, as the case's title suggests, the main data source is a mongo collection related to "IOHub Companies" providing a wide panoply of information on the companies. As complementary sources the following are used:

- A mongo collection, "Locations_pt", which allows replacing all generated ids present in the main source by the corresponding names (for districts, cities, and parishes);
- A mongo collection, "Jobs_pt", which is quite similar to the first one presented but applied to the jobs' context, in other words, it allows replacing the job's code present in the main source by the job's description;
- A mongo collection, "Cae_cirs", which also allows replacing a code for a description, but by CAE, i.e., economic activity code;
- An external JSON file, "Zip_codes_locations", acquired by merging three other different files retrieved from the Portuguese data central[40], in order to get a file composed by the two different parts that compose the zip code along with the corresponding city and the district. With that it is possible to fill the zip codes' dimension in a complete way and allow a most accurate validation of the zip codes in the database;

[40] http://centraldedados.pt/codigos_postais

- An external CSV file, "All_locations_pt", acquired by merging three other different files, two of them retrieved from the Portuguese data central and a third one[41] composed by a mapping between parishes' old names and their new ones, in order to get a file composed by parishes, along with their cities and districts. This serves the purpose of filling the locations' dimension in the most complete possible way.

5.3. DATA ANALYSIS

After getting access to the test data from IOTech, its full analysis was required in order to understand the meaning, type of data and value in the enterprise's context. So, in this section, it is possible to find this analysis concerning only the values that might have interest to this project context, even some of the following values might not be of any use because of some other problems, like all being null in the test file.

Notice that the test data are in JSON format, which is composed of objects, keys, and values. So, to make how the database is structured a little bit more visual, in an object matter, it is possible to find an example of this structure in Appendix 1 – IOHub Companies Database Structure.

Figure 47 is intended to clarify the importance of this analysis to the project as a whole. So, when looking at the figure, it is possible to see all data sources chosen to support this case study, already presented in this document, converging to the data processing API. Before they arrive to the API, this analysis is performed, allowing to define more precisely which transformations must be done inside the API that will result in the completion of the multidimensional model. Therefore, it becomes easy to understand the impact a good analysis will have on further work. A good analysis will lead to better data treatment without many avoidable twists, resulting in a more accurate multidimensional model and reporting results.

[41] http://smi.ine.pt/

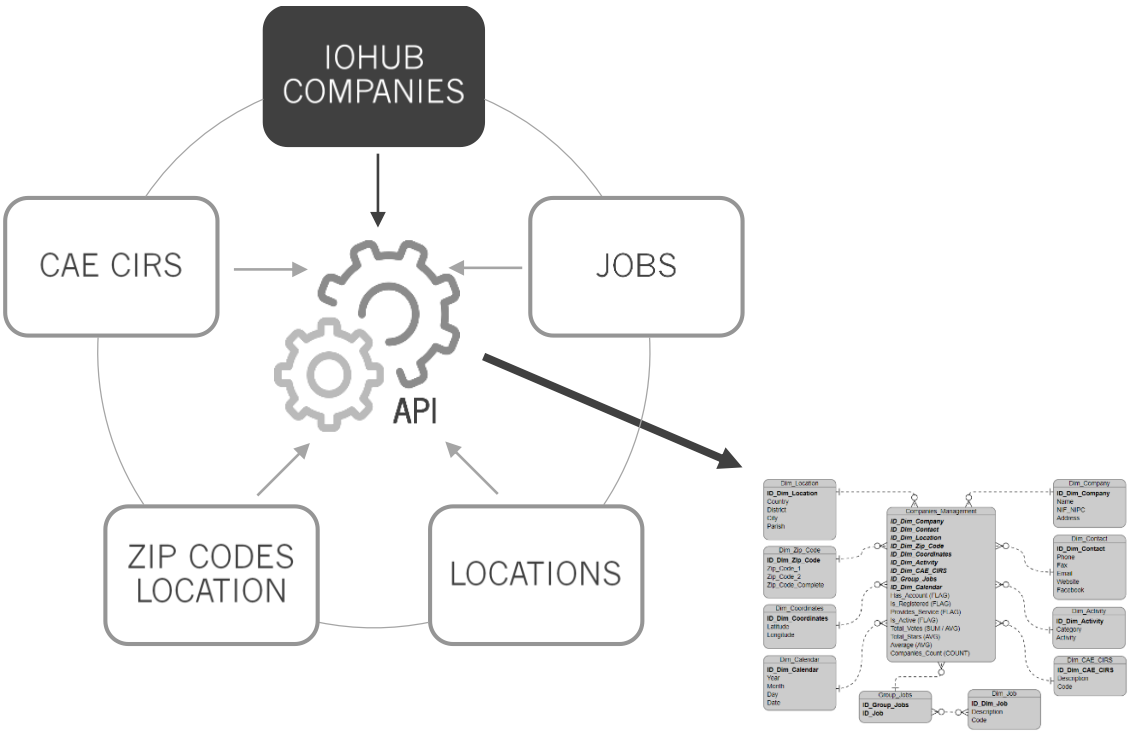


Figure 47 - Data Sources Analysis Result.

5.3.1. IOHUB COMPANIES COLLECTION

In the following table, it is possible to see a resume on the test data from the main data source analysis outcome by presenting the object, the context meaning, and brief errors and gaps detection. In this report, it is not presented the full extension of this database once it includes several objects completely irrelevant for this project and is most likely empty. For example, in the first row, it is presented the object "_id", which is varchar, and it's a simple index.

Table 11 - Data Analysis - IOHub Companies Collection.

OBJECTS				MEANING / ANALYSIS
_id				Simple index object with no value for the context (key). Although, since it is an "ObjectId" type of index, the register date into the database can be retrieved.
info	company	activity	description	Activity description associated with a company in the system. This object has some issues like several null values even though it exists in all database instances.
			category	Activity category associated with a company in the system. This object exists in all database instances but is populated mainly with null values.
			provides_service	This object indicates if this company provides or not a service. This object does not present issues.
		cae cirs	primary	Code CAE associated with a company's category of work. This object as an inconsequential ratio of 7% of null values. Another fact to be pointed is the different levels of activity categorization presented here. Those differences are not treated as an error.
			secondary	Code CAE secondary number associated with a company's category of work. This object is empty in all database instances. Given that and the fact that this does not increase the information quality, it is not used.
			description	Description of CAE's code work category associated. This object is empty in all database instances. Given that this object is not used, but this information is retrieved from another data source.
			text	Extra text for further explanations on the CAE's code work category associated. This object is empty in all database instances. Given that and the fact that this does not increase the information quality, it is not used.
		contact	phone	Company's phone number. This object has more than one-third of null instances but still is relevant for the context for what it is used.
			website	Company's website link. This object is mainly fulfilled with null values and is also inexistent in some instances of the database. Another kind of error found is related to the website link structure that is not valid. Regardless of the

OBJECTS				MEANING / ANALYSIS
				previous points, this object is used in the multidimensional model once in the final dataset, it is expected to find more complete data, and it might be relevant to keep.
			email	Company's email. Similarly, to others, this object is mainly fulfilled with null values but still present in all instances of the database. Another kind of error found is related to the email structure that is not valid. Regardless of the previous points, this object is used in the multidimensional model once in the final dataset, it is expected to find more complete data, and it might be relevant to keep.
			fax	Company's fax number. Despite the fact that, so far, this object only counts with null values, it is used once it has the potential to be a relevant set of information.
			facebook	Company's facebook link. Despite the fact that, so far, this object only counts with null values, it is used once it has the potential to be a relevant set of information.
		location	address	Company's address location. This object is present in all instances of the database and counts with an irrelevant amount of null values (around 0.3%).
			parish	Company's parish location. This object is not present in a few instances and has null values in some others. Furthermore, there are some inconsistencies once this object is supposed to store an id referring to the parishes' names stored in a different collection, but sometimes the names are stored here.
			city	Company's city location. This object is not present in a few instances and has null values in some others. Furthermore, there are some inconsistencies once this object is supposed to store an id referring to the cities' names stored in a different collection, but sometimes the names are stored here.
			region	Company's district/autonomous region location. This object is not present in a few instances and has null values in some others. Furthermore, there are some inconsistencies once this object is supposed to store an id referring to the districts'/autonomous regions' names

OBJECTS					MEANING / ANALYSIS
					stored in a different collection, but sometimes the names are stored here.
			country		Company's country location. This object has no errors or gaps to be reported.
			zip code	zc_1	Company's first number of zip code. This object can be found in all database instances, but in, approximately, 5% of them its value is null.
				zc_2	Company's second number of zip code. This object can be found in all database instances, but in, approximately, 5% of them its value is null.
			coordinates	lat	Company's latitude location. This object can be found in all database instances, but in, approximately, 3% of them its value is null.
				lon	Company's longitude location. This object can be found in all database instances, but in, approximately, 3% of them its value is null.
		begin_date			Date in which the company started its functions. This object has null values for all database instances. In the future, it should be considered to be part of the multidimensional model, but so far, it is completely irrelevant.
		commercial_name			Company's commercial name. This object has null values for all database instances. Given that and the fact that there is another object with similar information in the collection, this one is not considered.
		name			Company's name. This object has only one null value. Given its character, this object is used as an exclusion factor once it is illogical to keep a record of companies whose name is not known.
		slogan			Company's slogan. This object has null values for all database instances. Given that and the fact that it does not provide relevant information, this one is not considered.
		logo_url			Company's logo URL. This object has null values for all database instances. Given that and the fact that it does not provide relevant information, this one is not considered.

OBJECTS			MEANING / ANALYSIS
	rating_score	total_votes	Company's number of votes used to rate the company. This object has almost all instances filled with value "0", in this context this is comparable to a null situation. Still, this object has the potential to provide valid and relevant information for what it keeps its spot in the multidimensional model.
		total_stars	Company's number of stars with which the company is rated. This object has almost all instances filled with value "0", in this context this is comparable to a null situation. This object goes along with the previous one for what it is kept.
		average	Company's average rate. This one is a consequence of the two objects above for what it is also kept regardless of the huge amount of "0" values.
	name		Company's name. This object has null values for all database instances. Given that and the fact that there is another object with similar information in the collection, this one is not considered.
	nif_nipc		Company's NIF or NIPC, according to being a singular or collective person that owns the number. This object has only one null value what is completely irrelevant.
	img_url		Company's image URL. This object has null values for all database instances. Given that and the fact that it does not provide relevant information, this one is not considered.
	jobs		Group of jobs' ids involved in the company's activities. For approximately 12.5% of instances, this object is an empty array which is similar to a null case. Still, this object provides relevant information and therefore is applied to the multidimensional model.
other	valid_nif		Boolean indicator of a company's NIF or NIPC validity. For all instances, this object has "false" as the value which does not correspond to the truth and therefore is ignored, and a validation mechanism is developed.
	is_registered		Boolean indicator of the register state of a company. For all instances, this object has "false" value, but in the

OBJECTS	MEANING / ANALYSIS
	future, it is expected for it to be trustfully complete once it is an important fact in the context and for that, it is kept.
has_account	Boolean indicator of the existence of a company's account. For all instances, this object has "false" value, but in the future, it is expected for it to be trustfully complete once it is an important fact in the context and for that, it is kept.
active	Boolean indicator of a company's activity state. This object has no errors to be reported.
user_type	Indicator of user type, which in this dataset is always "Company". This object has no errors, but still, it is irrelevant for this context being for that dismissed.
register_date	Date in which the company was registered into the system. This object has null values for all database instances. In the future, it should be considered to be part of the multidimensional model, but so far, it will be "replaced" by the date retrieved from the "_id" object.
language	Language associated with the company's usage, which in this dataset is always "pt" (Portuguese). This object has no errors, but still, it is irrelevant for this context being for that dismissed.

5.3.2. CAE CIRS COLLECTION

In the following table, Table 12, it is possible to see a resume on the test data from CAE CIRS' data source analysis outcome by presenting the object, the context meaning, and brief errors and gaps detection. For example, in the first row, it is presented the object "_id", which is a simple index and is not used for that reason.

Table 12 – Data Analysis - CAE CIRS Collection.

OBJECT	MEANING / ANALYSIS
_id	Simple index object with no value for the context (key). This has no use in the context.
cae	Code CAE associated with a category of work. This object as no errors to be reported.
description	Description of CAE's code work category associated. This object as no errors to be reported.

OBJECT	MEANING / ANALYSIS
jobs	Group of jobs involved in the CAE's work category. For more than half of instances, this object is an empty array which is similar to a null case. Once it does not allow to complete the main dataset, it is ignored.

5.3.3. JOBS COLLECTION

In the following table, Table 13, it is possible to see a resume on the test data from jobs' data source analysis outcome by presenting the object, the context meaning, and brief errors and gaps detection. For example, in the first row, it is presented the object "_id", which is a simple index and is not used for that reason.

Table 13 - Data Analysis - Jobs Collection.

OBJECT	MEANING / ANALYSIS
_id	Simple index object with no value for the context (key). This has no use in the context.
id	Id of a job. This object as no errors to be reported.
name	Description of a job's id. This object as no errors to be reported.

5.3.4. LOCATIONS COLLECTION

In the following table, Table 14, it is possible to see a resume on the test data from locations' data source analysis outcome by presenting the object, the context meaning, and brief errors and gaps detection. For example, in the first row, it is presented the object "_id", which is a simple index and is not used for that reason.

Table 14 - Data Analysis - Locations Collection.

OBJECT	MEANING / ANALYSIS
_id	Simple index object with no value for the context (key). This has no use in the context.
name	Designation of a location (parish, city or district). This object has no errors to be reported.
type	The type of location should be considered the level of location: level one for the district, level two for the city and level three for the parish. This object has no errors to be reported.
descendants	This is a group of ids indicating all locations at which level is inferior to the current one. Some instances have this object corresponding to an empty array which is similar to a null situation, but

OBJECT	MEANING / ANALYSIS
	not all of them should be considered an error since all parishes have no descendants. This object is not considered.
ascendants	This is an id indicating the location at which level is superior to the current one. Some instances have this object has null, but not all of them are errors once all districts have no ascendants. This object is not considered.

5.3.5. ZIP CODES LOCATION FILE

In the following table, Table 15, it is possible to see a resume on the test data from zip codes locations' data source analysis outcome by presenting the object, the context meaning, and a brief errors and gaps detection. For example, in the first row, it is presented the object "district", which is stores the names of the districts and has no errors.

Table 15 - Data Analysis - Zip Codes Location File.

OBJECT	MEANING / ANALYSIS
district	This object stores the name of all districts and autonomous regions of Portugal. This object has no errors to be reported.
city	This object stores the name of all the cities of Portugal. This object has no errors to be reported.
zip_code_1	This object corresponds to the first part of a zip code correspondent to the city and district of the same instance. This object has no errors to be reported.
zip_code_2	This object corresponds to the second part of a zip code correspondent to the city and district of the same instance. This object has no errors to be reported.

5.4. MULTIDIMENSIONAL MODEL

In Figure 48, it is possible to see the multidimensional model that supports this project concerning the "IOHub Companies" case study and a brief description of what its components are and mean to this project. It is important to state that this model is shaped in a Star Multidimensional Model, that is why it has just one fact's table connected to all dimensions of the model presented.

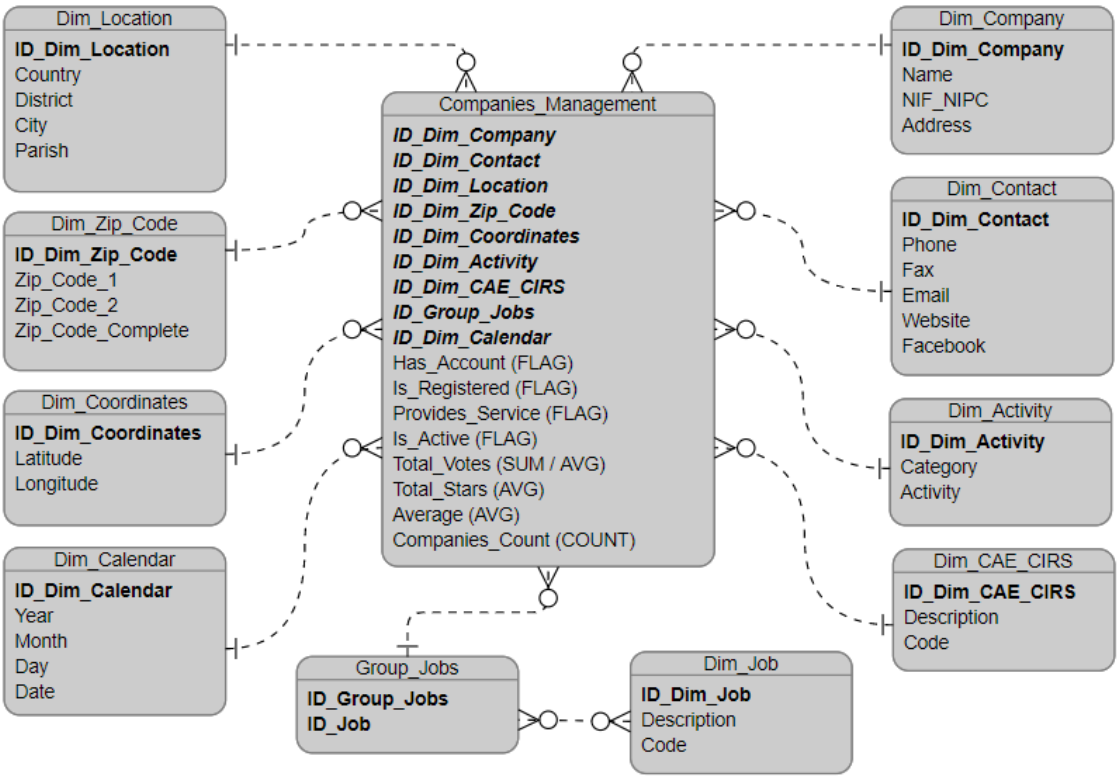


Figure 48 - Multidimensional Model - IOHub Companies Case Study.

5.4.1. DIMENSION LOCATION

This dimension appears with the goal of positioning a company in a matter of space, in other words, this dimension serves to answer questions such as "Where?".

For that, this dimension is composed of spatial fields like Country, District, City, and Parish. It is notable the existence of a hierarchy between those fields, being the most high-level location given by the Country field and the most specific by the Parish field.

Table 16 - Dimension Location Description.

FIELD NAME	DESCRIPTION	TYPE	EXAMPLE
ID_Dim_Location (SK, PK)	This is a meaningless primary key, for that called server key used uniquely with the goal of identifying a specific line.	Integer	1
Country	This field is responsible for giving information about which country is a company located.	String	Portugal

FIELD NAME	DESCRIPTION	TYPE	EXAMPLE
District	This field is responsible for giving information about in which district is a company district.	String	Porto
City	This field is responsible for giving information about which city is a company located.	String	Porto
Parish	This field is responsible for giving information about which parish is a company located.	String	Cedofeita

5.4.2. DIMENSION ZIP CODE

Similarly, to the previous dimension, this one aims of positioning a company in a matter of space, but this time a little more specific using for that the zip codes.

Thus, this dimension is composed of spatial fields like the different parts of a zip code and the complete version of it.

Table 17 - Dimension Zip Code Description.

FIELD NAME	DESCRIPTION	TYPE	EXAMPLE
ID_Dim_Zip_Code (SK, PK)	This is a meaningless primary key, for that called server key used uniquely with the goal of identifying a specific line.	Integer	1
Zip_Code_1	This field is responsible for giving information about the first number that composes a company's zip code.	String	4805
Zip_Code_2	This field is responsible for giving information about the second number that composes a company's zip code.	String	110
Zip_Code_Complete	This field is responsible for giving complete information about a company's zip code.	String	4805-110

5.4.3. DIMENSION COORDINATES

Similar to the previous dimensions, this one aims of positioning a company in a matter of space, in the most specific way by using for that coordinates.

Hence, this dimension is composed of spatial fields like latitude and longitude.

Table 18 - Dimension Coordinates Description.

FIELD NAME	DESCRIPTION	TYPE	EXAMPLE
ID_Dim_Coordinates (SK, PK)	This is a meaningless primary key, for that called server key used uniquely with the goal of identifying a specific line.	Integer	1
Latitude	This field is responsible for giving information about the first number that composes a company's zip code.	Double	40.985632
Longitude	This field is responsible for giving information about the second number that composes a company's zip code.	Double	-8.256358

5.4.4. DIMENSION CALENDAR

This dimension purpose is to position a company in a matter of time, that means answering questions such as "When?". Right now, this is used to identify when a company was inserted in the database but in the future could be used to identify other relevant time marks.

Therefore, this dimension is composed of fields like Day, Month, Year and Date. Then again, it is easy to identify a hierarchy between some of those fields, being Year the most general time location and Day the most specific one.

Table 19 - Dimension Calendar Description.

FIELD NAME	DESCRIPTION	TYPE	EXAMPLE
ID_Dim_Calendar (SK, PK)	This is a meaningless primary key, for that called server key used uniquely with the goal of identifying a specific line.	Integer	1
Year	This field is responsible for giving information about which year was a company inserted in the database.	Integer	2018
Month	This field is responsible for giving information about which month was a company inserted in the database.	Integer	01
Day	This field is responsible for giving information about which day was a company inserted in the database.	Integer	01
Date	This field is responsible for giving information about which date was a company inserted in the database.	Date	01-01-2018

5.4.5. DIMENSION COMPANY

This dimension aims to identify a company and some of its data. That means that this dimension is responsible for answering questions like “Who?”.

For that to be possible, the Company’s dimension is composed of fields like Name, NIF or NIPC and Address.

The reason for the address being positioned in this dimension and not in Location is because it would cause an unnecessary increase in the Location’s dimension. Here address, even identifying the spatial location of a company it is treated as an individual piece of information of a company.

Table 20 - Dimension Company Description.

FIELD NAME	DESCRIPTION	TYPE	EXAMPLE
ID_Dim_Company (SK, PK)	This is a meaningless primary key, for that called server key used uniquely with the goal of identifying a specific line.	Integer	1
Name	This field is responsible for giving information about the name of a company.	String	Nice Kids, Lda.
NIF_NIPC	This field is responsible for giving information about the NIF or NIPC of a company.	Integer	987654321
Address	This field is responsible for giving information about the address of a company	String	Rua Serra 700 Maia

5.4.6. DIMENSION CONTACT

This dimension aims to identify the contacts related to a company. That means that this dimension is responsible for answering questions like "How?", particularly, how to contact a company.

For that to be possible, the Contact's dimension is composed of fields like Phone, Fax, Email, Website, and Facebook.

Table 21 - Dimension Contact Description.

FIELD NAME	DESCRIPTION	TYPE	EXAMPLE
ID_Dim_Contact (SK, PK)	This is a meaningless primary key, for that called server key used uniquely with the goal of identifying a specific line.	Integer	1
Phone	This field is responsible for giving information about which phone number corresponds to a company.	Integer	912565692

FIELD NAME	DESCRIPTION	TYPE	EXAMPLE
Fax	This field is responsible for giving information about which fax number corresponds to a company.	Integer	252303400
Email	This field is responsible for giving information about which email corresponds to a company.	String	webmail@mercus.pt
Facebook	This field is responsible for giving information about which facebook link corresponds to a company.	String	https://www.facebook.com/uminhooficial/
Website	This field is responsible for giving information about which website link corresponds to a company.	String	www.digitosolutions.com

5.4.7. DIMENSION ACTIVITY

This dimension aims to identify the activity's field of a company. That means that this dimension is responsible for answering questions like "What?", particularly, what the company does.

For that to be possible, the Activity's dimension is composed of fields like Activity and Category. Between these fields, there is a hierarchy that can be defined, placing Category as the most general and Activity the most specific.

Table 22 - Dimension Activity Description.

FIELD NAME	DESCRIPTION	TYPE	EXAMPLE
ID_Dim_Activity (SK, PK)	This is a meaningless primary key, for that called server key used uniquely with the goal of identifying a specific line.	Integer	1
Activity	This field is responsible for giving information about the activity field in which the company actuates.	String	Instalação Elétrica

FIELD NAME	DESCRIPTION	TYPE	EXAMPLE
Category	This field is responsible for giving information about the activity category in which the company actuates.	String	Têxteis – Lar

5.4.8. DIMENSION CAE CIRS

Following the same idea of the last dimension presented, this one serves to identify the activity's field of a company according to CAE CIRS ("Classificação das Atividades Económicas Portuguesas por Ramo de Atividade"). That means that this dimension is responsible for answering questions like what the company does according to the CAE system.

For that to be possible, this dimension is composed of fields like Code and Description of CAE CIRS.

Table 23 - Dimension CAE CIRS Description.

FIELD NAME	DESCRIPTION	TYPE	EXAMPLE
ID_Dim_CAE_CIRS (SK, PK)	This is a meaningless primary key, for that called server key used uniquely with the goal of identifying a specific line.	Integer	1
Code	This field is responsible for giving information about the CAE code of the field in which the company actuates.	Integer	1491
Description	This field is responsible for giving information about the CAE description of the field in which the company actuates.	String	Apicultura a criação de abelhas

5.4.9. DIMENSION JOB

This last dimension serves to identify the jobs related to a company's activities. That means that this dimension is responsible for answering questions "How?", how is the activity realized, using which kind of resources?

For that to be possible, this dimension is composed of fields like Code and Description of a job.

Table 24 - Dimension Job Description.

FIELD NAME	DESCRIPTION	TYPE	EXAMPLE
ID_Dim_Job (SK, PK)	This is a meaningless primary key, for that called server key used uniquely with the goal of identifying a specific line.	Integer	1
Code	This field is responsible for giving information about a job code involved in certain activities.	Integer	101
Description	This field is responsible for giving information about a job description involved in certain activities.	String	Engenheiro

5.4.10. BRIDGE TABLE GROUP JOBS

This bridge table appears with the goal of mitigating a problematic many-to-many relationship between the facts table and the dimension job. This question arises once it is clear that, in order to handle its activities, a company might need more than one kind of specialist and, therefore, different match with the job's dimension.

So, this bridge table only includes the job's group id and the job id, which together represent a primary key for this table.

Table 25 - Bridge Table Group Jobs Description.

FIELD NAME	DESCRIPTION	TYPE	EXAMPLE
ID_Group_Jobs (SK, PK)	This is a server key, used uniquely with the goal of identifying a specific group in the case of bridge tables.	Integer	403
ID_Dim_Job (FK, PK)	This is a foreign key, once it is the primary key of the job's dimension. This one, combined with the id of the job's group represents the primary key of this table.	Integer	1

5.4.11. FACTS TABLE COMPANIES MANAGEMENT

The facts table is the table in which all the information spread along the multidimensional model converges by the mean of indexation. In other words, this table is composed of a set of ids that allow linking the lines in this table to the information in the different dimensions.

More than that, it is here that it is possible to find the facts that have study relevance for the organization. So, this table, beyond the diverse dimension's ids includes a bunch of facts such as: is registered, has an account, is active, total votes, average, total stars, provides service, and a counter.

Table 26 - Facts Table Companies Management Description.

FIELD NAME	DESCRIPTION	TYPE	EXAMPLE
ID_Dim_Location (FK, PK)	This is a foreign key that allows this table to relate to the location's dimension by its primary key.	Integer	1
ID_Dim_Zip_Code (FK, PK)	This is a foreign key that allows this table to relate to the zip code's dimension by its primary key.	Integer	1

FIELD NAME	DESCRIPTION	TYPE	EXAMPLE
ID_Dim_Coordinates (FK, PK)	This is a foreign key that allows this table to relate to the coordinates' dimension by its primary key.	Integer	1
ID_Dim_Calendar (FK, PK)	This is a foreign key that allows this table to relate to the calendar's dimension by its primary key.	Integer	1
ID_Dim_Company (FK, PK)	This is a foreign key that allows this table to relate to the company's dimension by its primary key.	Integer	1
ID_Dim_Contact (FK, PK)	This is a foreign key that allows this table to relate to the contact's dimension by its primary key.	Integer	1
ID_Dim_Activity (FK, PK)	This is a foreign key that allows this table to relate to the activity's dimension by its primary key.	Integer	1
ID_Dim_CAE_CIRS (FK, PK)	This is a foreign key that allows this table to relate to the CAE CIRS' dimension by its primary key.	Integer	1
ID_Group_Jobs (FK, PK)	This is a foreign key, that allows this table to relate to the group job's bridge table by the first part of its primary key, allowing to get the several second parts.	Integer	1
Is_registered	This is one of the facts that this table contains. This retrieves information on either a company is or not registered into the system.	Boolean	0
Has_account	This is one of the facts that this table contains. This retrieves information on either a company has or not account in the system.	Boolean	0

FIELD NAME	DESCRIPTION	TYPE	EXAMPLE
Is_active	This is one of the facts that this table contains. This retrieves information on either a company is or not active in the system.	Boolean	1
Total_votes	This is one of the facts that this table contains. This retrieves information on how many votes does a company has.	Integer	2
Total_stars	This is one of the facts that this table contains. This retrieves information on how many stars is a company rated.	Integer	10
Average	This is one of the facts that this table contains. This retrieves information on the average votes that a company obtained.	Double	2.5
Provides_service	This is one of the facts that this table contains. This retrieves information on if a company is a services provider or not.	Boolean	1
Companies_counter	This is one of the facts that this table contains. This is a simple count of companies to allow further aggregation analysis.	Integer	1

6. CONCLUSIONS

With this chapter's conclusion, it is possible to understand how the solution is designed and structured by looking at the high-level architecture presented in it. Furthermore, the notion of this project's contributions to the community becomes clearer when looking at it as a whole but understanding each and every one of its components.

Regarding the development layers and methods and materials chosen, it is possible to understand each path's turn selected to carry out the solution presented at the end of this dissertation, and to know which other possibilities were available in the market (or not) and that could be used in different realities and variations of this project.

Furthermore, a set of possibilities is mentioned, along with improvements proposed to be done in the future to complement the solution presented and redirect it according to the needs of any different case studies that might appear.

CHAPTER 5 – CONCLUSION

In this final chapter the project's conclusions are presented by offering some final considerations (where an overview over the developments is presented along with a cross-validation of the goals defined initially), as well as the limitations and difficulties faced throughout the project and their impact on the results obtained. Additionally, there are also some perspectives of what might be done to continue this thesis's work turning the solution into a strong and market-suitable one.

1. FINAL CONSIDERATIONS

This document's purpose was to describe the whole dissertation from the research phase until its practical results.

In a first approach, to contextualize this dissertation's project and the problematics involved, a literature selection was created in order to study some concepts that are at the base of this work, like data science, data analytics, real-time data, unstructured data and so on.

The first step, literature review, ends with a presentation of important concepts, their context and relevance for the case and the impact on the problematic of this dissertation and vice-versa. Furthermore, some solutions were proposed in order to recognize this theme's art state, as well as what has already been studied and successfully implemented. Based on the results of this literary process, it was easier to define a strategy to answer the questions presented at the beginning of this document.

After this, the actual solution development started with the gathering and exploration of technologies and tools according to the project's needs, for each level of the defined solution architecture, along with its development layers.

So, as proposed in the beginning of this project, an API capable of getting data from the document store, treat it according to the needs of structured and unstructured data was initially developed, over which an OLAP layer capable of querying data and passing the results onto a set of dashboards was built.

Well, between this OLAP layer and the web application, in order to add value to this project, even though it was not in the initial plan, a caching system was developed to allow some analysis even when the network connection is inexistent. This system is intimately linked with both OLAP

and visualization. Due to the innovation associated with this caching system, a previous patent request has been made.

Finally, the web application was mounted according to the predefined analysis (after the case study's exploration).

To understand the coverage of this project, in Table 27, some global metrics associated with this project are presented, from what was used to leverage it until what it includes.

Table 27 - Development Metrics.

METRIC	COUNT	METRIC	COUNT
Languages Used	2	Cubes	1
Libraries Used	5	Base Queries	22
Frameworks Used	4	KPIs	5
Databases	3	Charts	10
Tables	9	Dashboards	4
Treatment Functions	11	Modules	4

Based on this, and to understand the success of this project, it is relevant to check if the goals defined were achieved or not with the results obtained with this project. Therefore, in Table 28, it is possible to check the structuring goals for each main goal, as well as in which development layer they are present and if they were achieved or not.

Table 28 - Goals Cross-Validation.

MAIN GOAL	STRUCTURING GOALS	LAYER	STATE
Development of a prototype to consult information through different Dashboards in useful time over an OLAP structure.	Develop a Multidimensional Model.	Data*	Achieved
	Develop an OLAP layer.	Analysis	Achieved
	Develop a Web Paige to support the Dashboards.	Visualization	Achieved
	Develop a set of Dashboards.	Visualization	Achieved
	Analyze and Select Data.	Data*	Achieved

MAIN GOAL	STRUCTURING GOALS	LAYER	STATE
Development of an API, Application Programming Interface, to get data, prepare it and store it.	Treat the Data to be used.	Data	Achieved
	Develop mechanisms to handle Unstructured Data.	Data	Achieved
	Develop mechanisms to retrieve Information	Data	Achieved

* Done in the context of the case study being examined, using data provided by the enterprise IOTech, used as proof of concept.

At this point, it is more than appropriate to take a look back into the investigation's question presented in the beginning of this dissertation: "How can we process such amount of different kinds of data in order to extract relevant knowledge in useful time?", and state that this dissertation's output should answer it properly by achieving all goals defined to suit the question mentioned just before (as it can be seen in the Product's Backlog).

Looking at Table 28 is fair to assume that the project is being concluded successfully, all initial goals having been achieved. Of course, this does not mean the project is perfect and ready for market implementation, and for that in the last section of this chapter (Future Work) some improvements are suggested.

In order to understand the true potential of the solution described throughout the present document, a SWOT (Strengths, Weaknesses, Opportunities, and Threats) analysis is included in Table 29. This introduces, on the one hand, the strengths and opportunities that leverage the project, underlining its potential and value to the scientific community, and on the other hand, the implicit weaknesses and threats that could take down a part, or parts, of the solution designed.

With this it is possible to go deeper into the project's understanding, enabling the delineation of strategies to transform all weaknesses into strengths, and threats into opportunities, preventing the solution's decay, and potentiating its growth, improvement, and perseverance.

Table 29 - SWOT Analysis.

STRENGTHS	WEAKNESSES
• Innovation as a global solution; • Modularity perspective; • Integrated perspective; • Adaptability across different platforms; • Preforming offline analysis; • Allows connection to different types of databases.	• High times of data processing; • Only manual data refresh; • Regarding the case study in hands, no proper KPIs implemented; • The requirement of several external components to work properly.
OPPORTUNITIES	**THREATS**
• New technologies appearance that would allow mitigating current weaknesses; • Case studies with new data fronts to be explored; • A new analysis using different charts approaches; • The appearance of new data structures oriented to massive processing tasks; • Adapt the solution to function as a Software as a Service (SaaS); • Scalability; • Technologic evolution aiming real-time data and analytical processes improvement.	• New, similar or better solutions appearance; • Bad data quality; • Overload the system; • New technologies appearance that would implicate the superation on the current strengths; • Discontinuation and deprecation of needed modules; • The increasing complexity of the system's environment.

2. LIMITATIONS, RISKS, AND DIFFICULTIES

Much like every project, this dissertation has a set of limitations and difficulties associated with it, some of them directly related to the risks presented in Table 30.

An important element of the project's plan is risk management, because once the risks linked with the project are known, it is possible to plan actions to avoid any problems caused by them and actions to reduce their impact on the solution. Therefore, it is hence presented a complete risk list, the calculation of its impact and what can be done to mitigate the problems.

In Table 30, it is possible to see the list of risks associated with this dissertation and the respective estimated probability any of them happening, the impact that it has on the project and its severity (the result of the multiplication between probability and impact). Both probability and impact are measured using a scale from 1 till 5 (1 the being minimum value and 5 the maximum value, which means the higher the value, the higher the probability or the impact); this makes it

that severity is rated between 1 and 25 (1x1 till 5x5). More than this, it is possible to find at least one measure (mitigating action) to reduce the severity of each risk.

Table 30 - Risk List.

ID	RISK	MITIGATION ACTION	P[42]	I[43]	S[44]
1	Lack of experience.	Read articles and books related to the themes. Keep dialogue with mentors to solve some questions.	4	4	16
2	A wrong approach to the investigation topic.	Keep dialogue with mentors in order to have feedback on the path taken. Restructure research.	3	5	15
3	The high complexity of the project.	Increase the knowledge of tools and techniques to be used. Keep dialogue with mentors to solve some questions.	3	4	12
4	Lack of information on the topic of the dissertation.	Keep discussing the theme with the project mentors.	2	4	10
5	Non-compliance with the goals and/or expected results.	Delay the project delivery.	2	5	10
6	Difficulties using necessary tools.	Explore a bit more the tool previously. Search documentation and tutorials on it.	3	3	9
7	Data for testing with low quality.	Submit the data to a careful analysis and identify errors and inconsistencies in order to reduce or obliviate them.	3	3	9
8	Lack of comprehension of the data.	Talk with the project mentors in order to mitigate these questions and gather more information about it.	3	3	9
9	Bad planning of the tasks.	Identify and prioritize the main tasks and adjust the work plan.	2	4	8
10	Bad writing related to using non-maternal language.	Use grammar tools to help in some corrections.	2	3	6

[42] P – Probability

[43] I – Impact

[44] S – Severity

ID	RISK	MITIGATION ACTION	P[42]	I[43]	S[44]
11	Changes in the goals and expected results of the project.	Adjust the work plan, trying to minimize the impact of those changes.	1	5	5
12	Lack of comprehension of the project and/or its goals.	Talk with the project mentors in order to mitigate these questions and gather more information about it.	1	5	5
13	Bad communication with the mentors.	Define communication platforms.	1	4	4
14	The bad choice between methodologies available.	Previous study about the most relevant methodologies on the topic. Discuss the topic with mentors.	1	4	4
15	Machine malfunctions.	Use backup files into another available machine.	1	3	3
16	Loose files.	Keep backups of all work in different places (for example, flash drive and cloud)	1	2	2

The lack of knowledge on the technologies used, associated with the fact they are quite new in the market, for sure induced some extra complexity/difficulty to the project, demanding more study and attempt/error approaches, which took some more extra time that could be spent improving some features of the prototype presented. Well, this leads to another question, which is the lack of information on how to solve some problems, and how to link some of the technologies used. Both of these problematics, together with the first, fourth, and sixth risks, were handled through the search and reading of documents, and in the lack of such possibilities, other approaches were performed, such as joining group chats dedicated to some technologies, like Cube.js, allowing to ask questions and exchange information with more experienced people in that area).

The first limitation faced through this project is related to the processing capacity available when it was time to execute the data processing, which translated into a long process. To get around this problem, different data structures and modelling processes were tested but still the solution was not optimal. It might become less of a problem when executed in an actual server instead of a personal computer.

A relevant limitation to this project is related to the data sources' quality, more specifically, related to the main data source. This impacts the consistency of the dashboards presented in matters of the defined KPIs. In other words, based on the attributes of the data source collection, it is possible to define some relevant KPIs, but that is a problem when the time comes to implement

them; the data are fictitious and have a lot of relevant fields with no information to be used. That mustn't be a problem any longer as soon as the database starts to be filled up with real data, allowing the real implementation of the defined KPIs.

3. FUTURE WORK

Even though this dissertation is successfully concluded, exceeding the expectations, according to the company with which it is involved, the work for this project's development must not end this way, for there are a lot of possibilities to increase its value and to turn it into a strong and solid solution to the problem being studied.

Therefore, some improvements have already been considered, such as:

- A more relevant and founded KPIs' definition according to the case study, as soon as the data stored in the database becomes more consistent and reliable;
- A new dashboards' approach, such as the inclusion of GeoJSON maps to establish areas in the map, dynamic filters by map or other charts' clicks, etc.;
- As already mentioned in this document, an automatic refresh mechanism must be included concerning the ETL process, allowing time and mode configurations (time interval to redo the process, and if the data warehouse must be entirely cleaned, as it is usually the case, or if it must follow some update check to clean and load the new data);
- Following the logic of the last topic, it becomes relevant to include proper mechanisms to handle slowly changing dimensions;
- Orient the solution to be multiuser on a basis Software such as a Service (SaaS). In other words, the solution must be prepared with a similar structure, no matter which user is connected, though the database that supports user experience must be unique according to the user logged in. So, by the time a user has access to the application, there must be a mechanism to load the dashboard components linked to the corresponding account and connect them with the respective database.

REFERENCES

Angelov, B., & Rao, B. (2008). Pervasive Information Systems Value Chain: A Services Perspective. *International Journal of Innovation and Technology Management* (pp. 1-24). New York: World Scientific Publishing Company.

Baesens, B. (2014). *Analytics in a Big Data World.* New Jersey: Wiley.

Bose, B. (2016, December 27). *Top 10 Data Analytics Tools*. Retrieved from Digital Vidya: https://www.digitalvidya.com/blog/data-analytics-tools/

Brynjolfsson, E., Hitt, L. M., & Kim, H. H. (2011, April 24). *Strength in Numbers: How does data-driven decision making affect firm performance?* Retrieved from Social Science Research Network: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=1819486

Burd, G. (2011). NoSQL. *Sysadmin*, 5-12.

Cady, F. (2016). *The Data Science Handbook.* Washington: Wiley.

Cattell, R. (2011). Scalable SQL and NoSQL Data Stores. pp. 12-27.

Cervone, H. F. (2011). Understanding agile project management methods using SCRUM. *OCLC Systems & Services: International digital library perspectives*, 18-22.

Chaudhuri, S., & Dayal, U. (1997, March). An Overview of Data Warehousing and OLAP Technology. pp. 65-74.

Chodorow, K. (2013). *MongoDB The Definitive Guide.* United States of America: O'Reilly Media.

Chouder, M. L., Rizzi, S., & Chalal, R. (2017, November 21). EXODuS. *Exploratory OLAP over Document Stores* , pp. 45-57.

Chu, E., Baid, A., Chen, T., Doan, A., & Naughton, J. (2007, September 7). *A Relational Approach to Incrementally Extracting and Querying Structure in Unstructured Data*, pp. 23-28.

Codd, E. F., Codd, S. B., & Salley, C. T. (1993). *Providing OLAP (On-line Analytical Processing) to User-Analysts: An IT Mandate.* Codd & Associates.

Columbus, L. (2016, November 27). Retrieved from Forbes: https://www.forbes.com/sites/louiscolumbus/2016/11/27/roundup-of-internet-of-things-forecasts-and-market-estimates-2016/#130f4040292d

DATAFLAIR. (2018, December 3). *Advantages and Disadvantages of Python Programming Language*. Retrieved from Data Flair: https://data-flair.training/blogs/advantages-and-disadvantages-of-python/

Davies, N., & Clinch, S. (2017). Pervasive Data Science. In N. Davies, & S. Clinch, *Pervasive Computing* (pp. 50-58). IEEE CS.

Dehdouh, K. (2016). Building OLAP Cubes from Columnar NoSQL Data Warehouses. pp. 166-179.

Dubey, R. (2018, August 1). *What is big data intelligence?* Retrieved from Quora: https://www.quora.com/What-is-big-data-intelligence

Ernst & Young LLP. (2011). *Digital data opportunities: Using insight to drive relevance in the digital world.* Retrieved from Ernst & Young LLP: https://www.ey.com/Publication/vwLUAssets/Digital_data_opportunities/$FILE/EY_Digital_data_opportunities.pdf

Füser, K., & Georgi, B. (2013). *Data intelligence: using data to help your decision-making.* Retrieved from Ernst & Young LLP: https://www.ey.com/Publication/vwLUAssets/EMEIA_FAAS_Data_Intelligence/$FILE/EMEIA_FAAS_Data_Intelligence.pdf

González, S. M., & Berbel, T. R. (2014). Considering unstructured data for OLAP: a feasibility study using a systematic review. *Revista de Sistemas de Informação da FSMA*, 26-35.

Hammes, D., Medero, H., & Mitchell, H. (2014, March 22). Comparison of NoSQL and SQL Databases in the Cloud. *NoSQL and SQL Databases*.

Hashem, I. A., Yaqoob, I., Anuar, N. B., Mokhtar, S., Gani, A., & Khan, S. U. (2014, June 11). The rise of "big data" on cloud computing. *Review and open*.

Hurwitz, J., Nugent, A., Halper, F., & Kaufman, M. (2013). *Big Data for Dummies.* New Jersey: John Wiley & Sons, Inc.

Jones, S. (2005). Toward an Acceptable Definition of Service. *IEEE Software*, 87-93.

Kanimozhi, K. V., & Venkatesan, M. (2015). Unstructured Data Analysis - A Survey. *International Journal of Advanced Research in Computer and Communication Engineering*, 223-225.

Kotecha, B. H., & Joshiyara, H. (2017). A Survey of Non-Relational Databases with Big Data. *International Journal on Recent and Innovation Trends in Computing and Communication*, 143-148.

Krishnan, K. (2013). *Data Warehousing in the age of Big Data.* ELSEVIER.

Kurkovsky, S. A. (2008, January). Pervasive computing: Past, present and future.

Lin, Y., Jun, Z., Hongyan, M., Zhongwei, Z., & Zhanfang, F. (2018). A Method of Extracting The Semi-structured Data Implication Rules. *8th International Congress of Information and Communication Technology* (pp. 706-716). China: ELSEVIER.

Liu, J., Dong, X., & Havely, A. (2006, June 7). *Answering Structured Queries on Unstructured Data*.

Liu, Z., Jiang, B., & Heer, J. (2013). imMens: Real-time Visual Querying of Big Data. *Eurographics Conference on Visualization (EuroVis).* Blackwell Publishing Ltd.

Mansmann, S., & Scholl, M. H. (2007, December). Extending the Multidimensional Data Model to Handle Complex Data. *Journal of Computing Science and Engineering*, 125-160.

Mansmann, S., Rehman, N. U., Weiler, A., & Scholl, M. H. (2012, November 12). *Discovering OLAP Dimensions in Semi-structured Data*, pp. 9-16.

Meirelles, D. S. (2006). O Conceito de Serviço. *Revista de Economia Política*, 119-136.

Mitreva, E., & Kaloyanova, K. (2013, September). *NoSQL Solutions to Handle Big Data*.

Nedelcu, B. (2013). Business Intelligence Systems. *Database Systems Journal vol. IV, no. 4*, 12-20.

OData. (2017, December 5). *OData - the best way to REST*. Retrieved from OData: https://www.odata.org/

Padhy, R. P., Patra, M. R., & Satapathy, S. C. (2011). RDBMS to NoSQL: Reviewing Some Next-Generation Non-Relational Database's. *International Journal of Advanced Engineering Sciences and Technologies 11*, 15-30.

Peffers, K., Tuunanen, T., Rothenberger, M. A., & S. Chatterjee. (2007). A design science research methodology for information systems research. *Journal of management information systems*, v. 24, p. 45-77.

Priebe, T., & Pernul, G. (2003). Ontology-based Integration of OLAP and Information Retrieval. *14th International Workshop on Database and Expert Systems Applications.* Prague: IEEE.

Proschool. (2018, May 8). *Top 10 Data Analytics Tools*. Retrieved from Proschool: https://www.proschoolonline.com/blog/top-10-data-analytics-tools/

Provost, F., & Fawcett, T. (2013). Data Science and its Relationships to Big Data and Data-Driven Decision Making. In F. Provost, & T. Fawcett, *Big Data.* Mary Ann Liebert, Inc.

Rai, S. (2016, August 22). *What is data intelligence?* Retrieved from Quora: https://www.quora.com/What-is-data-intelligence

Ravat, F., Teste, O., Tournier, R., & Zurfluh, G. (2007). A conceptual model for multidimensional analysis of documents. *International Conference on Conceptual Modeling* (pp. 550-565). Auckland, New Zealand: Springer.

Rossiter, D. G. (2012, August 12). Introduction to the R Project for Statistical Computing for use at ITC.

Rusu, O., Halcu, I., Grigoriu, O., Neculoiu, G., Sandulescu, V., Marinescu, M., & Marinescu, V. (2013, January). *Converting unstructured and semi-structured data into knowledge*.

Sareen, P., & Kumar, P. (2015). NoSQL Database and its comparison with SQL Database. *International Journal of Computer Science & Communication Networks*, 293-298.

Scheps, S. (2008). *Business Intelligence for Dummies.* Indianopolis: Wiley Publishing Inc.

Scholz, J. (2011). Coping with Dynamic, Unstructured Data Sets – NoSQL a Buzzword. *International Conference on Urban Planning, Regional Development and Information*, (pp. 121-129).

Schwaber, K., & Sutherland, J. (2017, November). The Scrum Guide. *The Definitive Guide to Scrum:*.

Simmhan, Y., & Perera, S. (2016, October 11). Big Data Analytics Platforms for Real-time Applications in IoT. In E. S. Pyne, B. P. Rao, & S. Rao, *Big Data Analytics: Methods and Applications.* India: Springer.

Sint, R., Schaffert, S., Stroka, S., & Ferstl, R. (2009, January). Combining Unstructured, Fully Structured and Semi-structured Information in Semantic Wikis.

Wolff, G. (2014). Big Data and SP Theory of Intelligence. *IEEE*, 301-315.

Zikopoulos, P. C., Eaton, C., Deroos, D., Seutsch, T., & Lapis, G. (2012). *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data.* Mc Graw-Hill.

APPENDIX 1 – IOHUB COMPANIES DATABASE STRUCTURE

Key	Type
(1) ObjectId("5c38cc5129bc810fec678e4d")	Object
_id	ObjectId
info	Object
company	Object
activity	Object
description	String
category	Null
provides_service	Boolean
keywords	Array
images	Array
cae_cirs	Object
primary	Int32
secondary	Array
description	Null
text	Null
contact	Object
phone	String
fax	Null
email	String
website	Null
facebook	Null
location	Object
coordinates	Object
lat	Double
lng	Double
zip_code	Object
zc_1	String
zc_2	String
address	String
city	String
region	String
parish	String
country	String
begin_date	Null
commercial_name	Null
name	String
slogan	Null
logo_url	Null
rating_score	Object
total_votes	Int32
total_stars	Int32
average	Int32
name	Null
nif_nipc	Int32
img_url	Null
jobs	Array
other	Object
valid_nif	Boolean
is_registered	Boolean
has_account	Boolean
active	Boolean
user_type	String
register_date	Null
language	String
id	String
__v	Int32
package	Object
current	Object
expires_on	Null
starts_on	Null
duration	Int32
type	String
next	Object
expires_on	Null
starts_on	Null
duration	Int32
type	String
suspended	Object
expires_on	Null
starts_on	Null
duration	Int32
type	String
days_left	Int32

Figure 49 - IOHub Companies Database Structure.

APPENDIX 2 – TEST CASES

In order to develop a RESTful Python API capable of dealing with unstructured data and model it to reach multidimensional models that would give proper support to the use cases over a table, it was necessary a previous delicate study of which structure would suit the pre-requisites more widely. Therefore, five different cases were implemented and tested according to code cleanliness, comprehensive data structure, data fixedness in the system to the following operations, less time as possible since data arrival and its loading, different data inputs support (structured and unstructured), and scalability proclivity.

1. MODELLING BASED ON ARRAYS

The first attempt was based on modelling taking in its base a set of arrays, in order words, all elements that would compose the model would be stored in arrays, that each element corresponds to a line stored in a dictionary. To facilitate the visualization of this structure it is here presented, in Figure 50, a schema composed by the model entities, where the data came from, where it goes, and which structures are adopted in each step along the way.

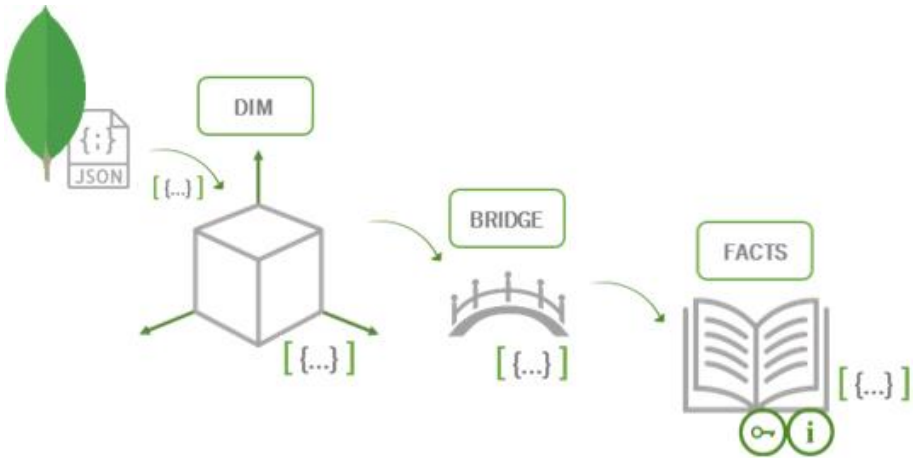


Figure 50 - Structure with Modeling based on arrays.

So, looking to Figure 50 it is possible to see that the data come from two specific different types of sources: MongoDB collections and JSON files. From there, these data are read into arrays that support the following operations' data requirements. Next to it, are displayed three different

"phases" labelled as "DIM", that stands for dimensions, "BRIDGE", that stands for bridge tables in the case it exists, and finally "FACTS", that stands for facts' tables.

All three of them are linked unidirectionally as the arrows suggest, this means that data flows towards the facts' tables and never the other way around. So, first of all, the data that is used to fill the different dimensions are joined, filtered, and properly treated and just then loaded into the different arrays that correspond to it.

Completed this task, the dimensions necessary to build up bridge tables are called through their arrays and iterated over along with other needed sources. Finished the grouping task the bridge tables are loaded into their own arrays.

Finally, and similarly to the previous "phase", it is time to deal with facts' tables. For that, all external data sources are called along with all arrays corresponding to each dimension and bridge table. In order to guarantee that the external data are comparable to the data stored in the dimensions and bridge tables, it must be submitted to a similar treatment to the data in other tables' arrays and some consistency-driven treatments.

In due time, each table array is iterated over in order to find a match and fill the id corresponding to that table straightaway. All ids identified and all facts retrieved from a data source or calculated, it is time to add a new dictionary into the facts' table array, were it stays as long as the system is running.

2. MODELLING BASED ON DATAFRAMES

This second test consisted on migrating the model from arrays to pandas dataframes, by other means, all elements that would compose the model would be stored in dataframes, that each element corresponds to a dataframe index. To facilitate the visualization of this structure it is here presented, in Figure 51, a schema composed by the model entities, where the data came from, where it goes, and which structures are adopted in each step along the way.

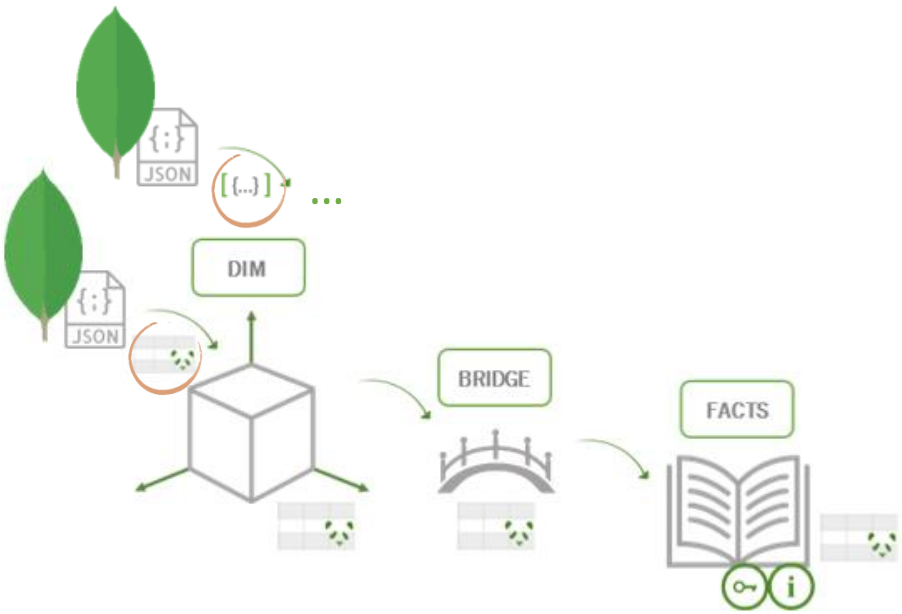


Figure 51 - Structure with Modeling based on dataframes.

Given the similarity between data and flow structure, from now on, the focus it is on the data structures changes and its impact. Another thing to notice is that in this section there are two different sub-trials, one "mixing" arrays with dataframes, and another one based uniquely on dataframes.

Like in the first attempt, in the first sub-trial, data are read into arrays that support the following operations' data requirements. During the second sub-trial, data was directly called into pandas dataframes and from there always managed like that. Between these two similar approaches, there was no advantage in selecting any of them over the other.

The structure followed to load the several dimensions that compose the model was basically the same. The main difference presents itself while loading both the bridge and the facts table. That is given by the fact that now it is possible to query directly the dataframe instead of "running" through all its elements to find what is desired and, therefore, get the needed ids in a simpler way.

3. MODELLING BASED ON NoSQL QUERIES

This approach follows the treatment flow exactly like the first time in pair with a staging area (S.A.). To facilitate the visualization of this structure it is here presented, in Figure 52, a schema composed by the model entities, where the data came from, where it goes, and which structures are adopted in each step along the way.

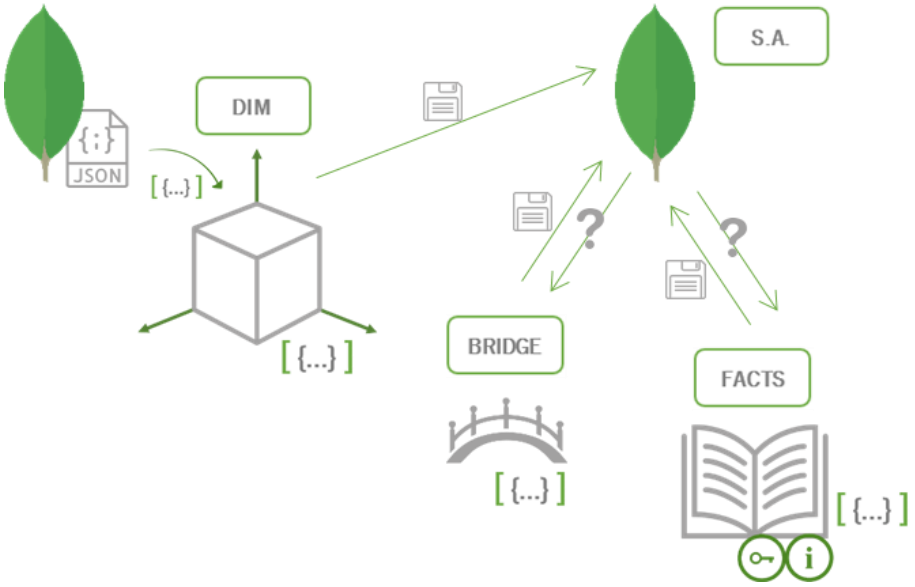


Figure 52 - Structure with Modeling based on NoSQL Queries.

Like the previous schemas, the flow is quite similar, but the querying component is strongly affected. To go through this explanation, it is important to understand how the staging area is organized. Therefore, in Figure 53, it is presented the staging area's structure, for the case study of IOHub Companies, where it is possible to gauge that each model table is represented by a different collection.

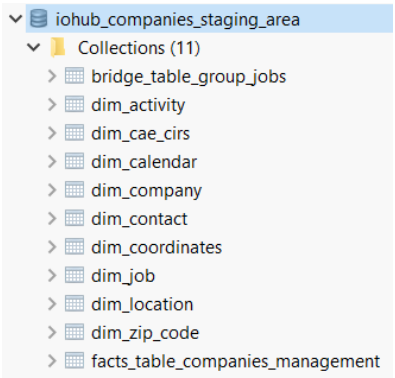


Figure 53 - Mongo DB Staging Area Structure – IOHUB Companies.

Well, considering the dimensions treatment, it is plausible to say that the job is done exactly in the same way as in the first attempt. The difference here is in the moment of loading the data. While in the arrays'-based modelling the dimensions are loaded in arrays' format, here, after all, the process is concluded (using arrays), the data are loaded into a Mongo DB collection present on the staging area.

In the two next "phases", things change a little bit once instead of iterating over dimensions' arrays, data are retrieved directly from Mongo DB using mongo queries for that. In other words, as soon as the treatment on dimensions side is concluded, data are sent to the staging area and in due time is queried in order to retrieve the necessary ids to fulfil both bridge table and facts table.

Well, that became too expensive in matters of time given the uncountable database calls. This problem relays on the fact that above all during facts' table loading, for each line, it is made one database query per each dimension, overloading the database processing which costs a lot of time.

4. MIXED MODELING WITH ARRAYS, DATAFRAMES AND UNSTRUCTURED STORAGE

This fourth attempt follows the treatment flow exactly like the previous one also in pair with a staging area (S.A.). To facilitate the visualization of this structure it is here presented, in Figure 54, a schema composed by the model entities, where the data came from, where it goes, and which structures are adopted in each step along the way.

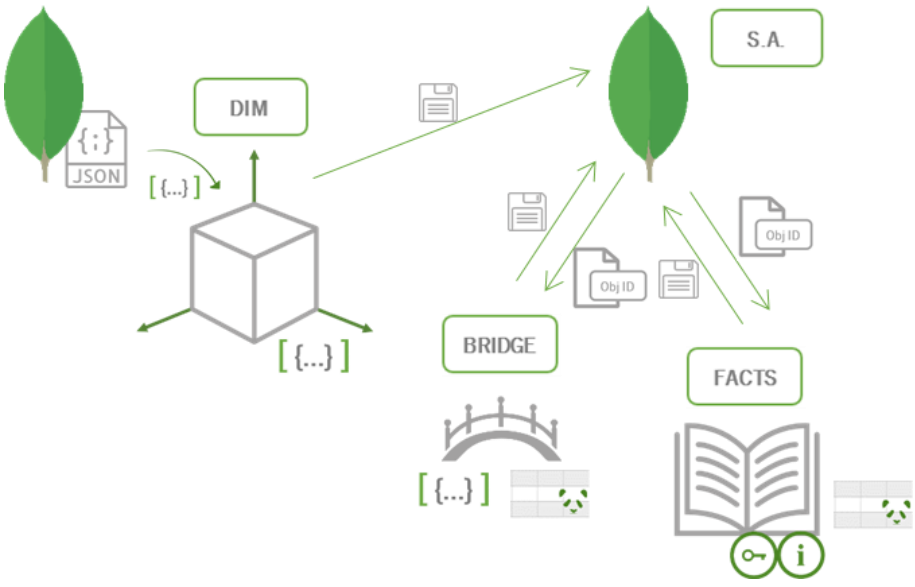


Figure 54 - Structure with Mix Modeling and Unstructured Storage.

First of all, the dimensions are loaded, and here there is no difference from the previous attempt. The difference here starts with the bridge table loading, which makes use of arrays and

dataframes for better coordination and performance. So, in order to load this table, first, data are pulled from the staging database (to avoid systematic queries and overload) into arrays. From there, groups are formed by successive iterations over the array and finally are outputted as a pandas dataframe, processed and then loaded into the staging area.

Similarly to the bridge table, to load the facts table, all data are pulled from the database, but this time directly into dataframes and from there queried to identify connections with the preprocessed data injected here. Only, in the end, all data are loaded into the staging area.

This solution revealed itself as less expensive but still with a few gaps to be solved and a considerable margin of improvement.

5. MIXED MODELING WITH ARRAYS, DATAFRAMES AND STRUCTURED STORAGE

This last attempt follows the treatment flow exactly like the previous one also in pair with a staging area (S.A.), but this time instead of using Mongo DB to do so, a structured database, MySQL, was used. To facilitate the visualization of this structure it is here presented, in Figure 55, a schema composed by the model entities, where the data came from, where it goes, and which structures are adopted in each step along the way.

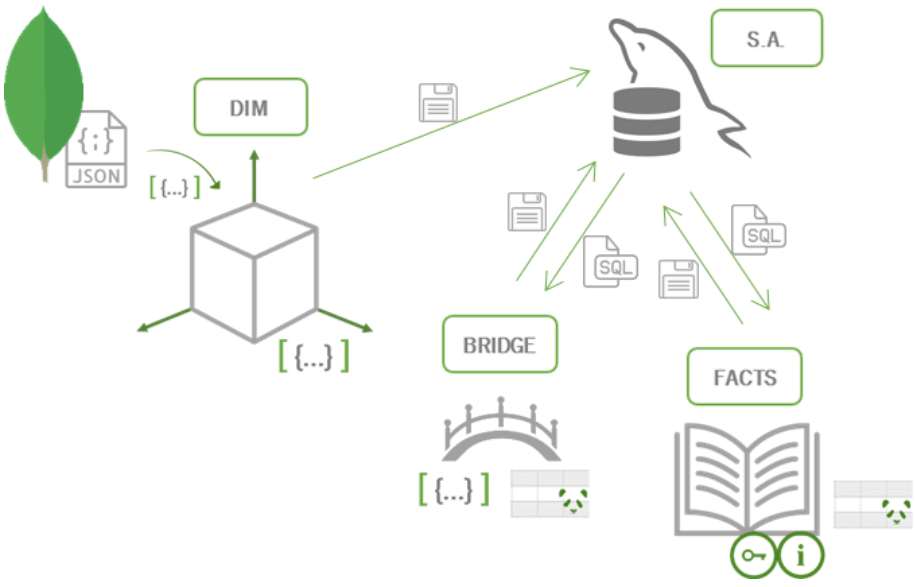


Figure 55 - Structure with Mixed Modeling and Structured Storage.

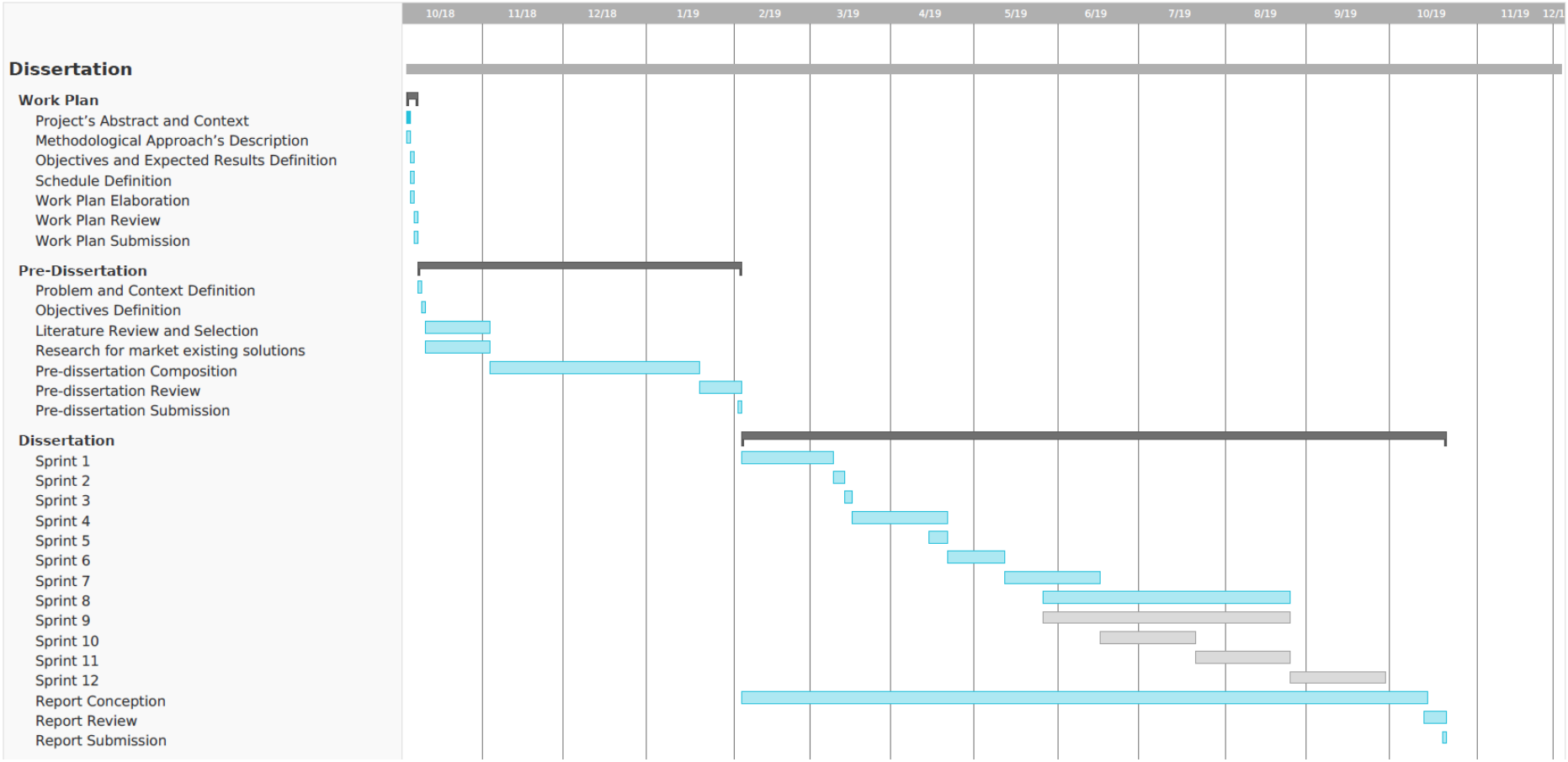
As it is understandable by looking to the schema of Figure 54 and the one in Figure 55, the only modification made is related with the kind of queries executed, in the previous trial they were NoSQL, and now they are SQL, which means that data was submitted into a structuring process. In other words, the unstructured data used as an input to this process was adulated in a manner that would fit a proper structure and stored in a MySQL database.

The question that emerges here is "Would the process be quicker?". Well, the answer is no, it would not, but increases the solution possibilities in another valuable way and allows the speed problem to be bypassed. So, by implement this solution the biggest advantage is related to the facility to support different types of data sources and improving the guarantees when this data feeds the OLAP layer (more uniformity guarantees more accurate results ahead).

Basically, what is being proposed with this alteration is a solution that can receive structured, semi-structured and unstructured data and according to its type react. This means that if the data received is structured, then the processing layer would be closer to the usual ETL made by the majority. Although, if the data became in stranger forms the processing layer would be, not only responsible for the ETL process but also to format and give structure to this data.

But how is the matter of this process speed solved? Well, once this question was a constant problem in all the proposed approaches and they were not capable of giving a pleasant answer to it, now the question is not how to solve but how to work around it. So, inevitably at the instant of creation of the model, it takes a considerable amount of time to process all the data until that moment, but from there the process does not take all that data into consideration but only the data added later or that were posteriorly edited. To do so it is used a database model that counts with creation and modification times, and then the processing layer is responsible for filtering which data are new and which data was edited and needs to be reprocessed. From there, the flow is followed in the normal way. This way the processing time would be drastically decreased after the model creation.

APPENDIX 3 – GANTT'S DIAGRAM



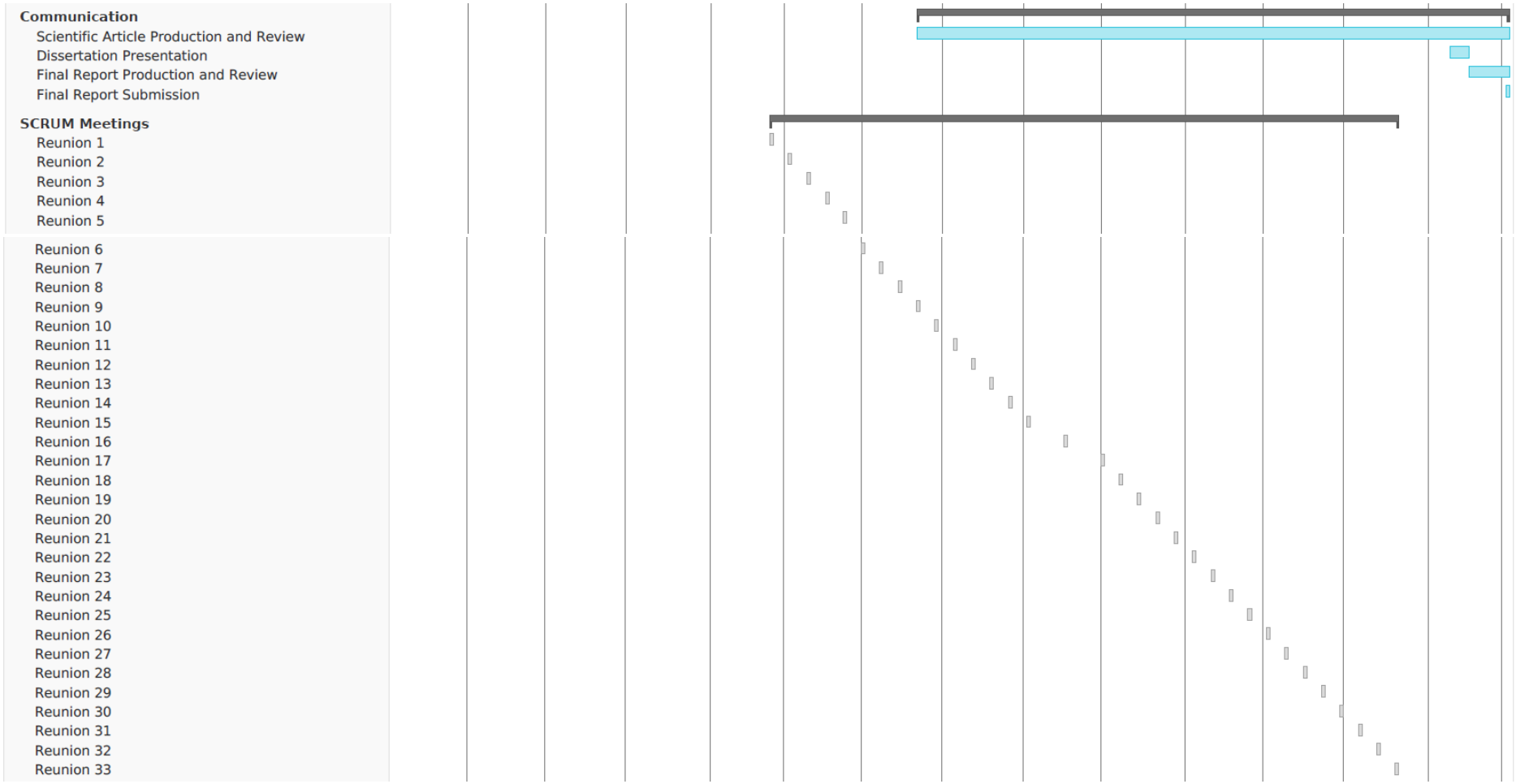


Figure 56 – Gantt's Diagram.

APPENDIX 4 – SCIENTIFIC PUBLICATION: TOWARDS THE DEVELOPMENT OF A DATA SCIENCE MODULAR SOLUTION

Authors: Gisela Fernandes, Filipe Portela, Manuel Filipe Santos

Conference: The 7th International Conference on Future Internet of Things and Cloud. Istanbul, Turkey. CPS. (2019)

State: Published

Abstract: With the technological progress that has been happening in the last few years, and now with the actual implementation of the Internet of Things concept, it is possible to observe an enormous amount of data being collected each minute. Well, this brings a problem along the way: "How can we process such amount of data in order to extract relevant knowledge in useful time?". That's not an easy issue to solve because most of the times it is needed to deal not just with tons of data but with different kinds of it which makes the problem even more complex. Therefore, the focus of this paper is to present a possible solution to this problem and try to give it a modular solution, adaptable to different realities, multi-schemas (structured and unstructured), using recent technologies and that allow users to access information where and when they wish.

Keywords: unstructured data, data warehouse, OLAP, ETL

APPENDIX 5 – SCIENTIFIC PUBLICATION: PWA AND PERVASIVE INFORMATION SYSTEM – A NEW ERA

Authors: Gisela Fernandes, Filipe Portela, Manuel Filipe Santos

Conference: 6th Pervasive Information Systems (WorldCist 2020)

State: Accepted

Abstract: Nowadays, and increasingly, the users' demand over the applications requires them to be a lot more flexible, adaptable, capable of being executed over different operational systems. This happens according to the need of accessing those applications no matter where, when, nor what they use to do so. Well, this is the basis under the concept of Pervasive Information Systems (PIS). But, how can such complex thematic be handled? How can an application be developed towards global usage? A new development methodology as been arising, Progressive Web Application (PWA), by mixing the web pages with the mobile applications world. So, in a nutshell, PWAs appear as a concretization of what is PIS concept. This article aims to explore this thematic and provide a few insights on how to develop a PWA.

Keywords: Progressive Web Applications (PWA), Pervasive Information System (PIS), multi-platform

APPENDIX 6 – SCIENTIFIC PUBLICATION: HOW TO BUILD A PWA – A PRACTICAL CASE STUDY

Authors: Gisela Fernandes, Olivério Sousa, Filipe Portela, Manuel Filipe Santos

Journal: MDPI Information or Future Internet

State: In submission

Abstract: Form a few years back, the need of accessing applications no matter where, when, nor what they use to do so, has been increasing exponentially. This comes associated with the concept of Pervasive Information Systems (PIS), that pushes the applications to be more flexible, adaptable, and capable of being executed over different operational systems. In order to address these thematics, a new web-based development methodology has been arising, Progressive Web Application (PWA), aiming to be applied across both web and mobile applications' world. In a synthetic and simplified way, it is plausible to say that PWAs appear as a concretization of what is PIS concept. This article aims to be an easy to follow guide on how to build a simple PWA (a cross-platform application that works regardless internet connectivity), taking on as a start point its architecture explanation. To concretize this matter an example of implementation is presented.

Keywords: Progressive Web Apps (PWA), pervasive computing, service worker, application shell, web app manifest

APPENDIX 7 – PREVIOUS PATENT APPLICATION

Authors: Gisela Fernandes, Filipe Portela

State: In submission

Abstract: A model, an architecture, and web-based solution aiming to provide Data Science as a Service (DsaaS), by retrieving information from heterogeneous data sources, processing it, and passing it through an analytical layer, towards a visualisation multi-platform application independent from network connection at all times. This might be applied in any context in which data plays a relevant role, by other means, it is not a closed solution. What is being proposed is a global solution in the area of data science that can be used in whichever business field who might need it.